IBM Cúram Social Program Management

# Cúram Case Audits Developers Guide

*Version 6.0.5*

IBM Cúram Social Program Management

# Cúram Case Audits Developers Guide

*Version 6.0.5*

**Note**
Before using this information and the product it supports, read the information in "Notices" on page 21

**Revised: May 2013**

This edition applies to IBM Cúram Social Program Management v6.0 5 and to all subsequent releases unless otherwise indicated in new editions.

Licensed Materials - Property of IBM.

# Contents

# Figures

# Tables

# Chapter 1. Introduction

## 1.1 Purpose

The purpose of this guide is to outline the available customization options for the Case Audits component and to provide instructions on how to implement these customizations.

## 1.2 Audience

This guide is intended for developers and architects intending to implement an auditing solution by customizing Cúram Case Audits.

## 1.3 Prerequisites

Before reading this guide the reader should be familiar with the Cúram Case Audits Guide. The reader should also be familiar with Google Guice.

## 1.4 Chapters in this Guide

The following list describes the chapters within this guide:

**Registering a New Algorithm**
This chapter describes how to add a new algorithm which provides a new method of producing a random sample of cases for audit.

**Utilizing Dynamic Selection Queries**
This chapter describes how to use Dynamic Selection Queries to generate a list of cases for audit.

**Configuring Selection Queries**
This chapter describes the configuration options available for Selection Queries.

**Case Audits Web Service**
This chapter provides a brief overview on how case data from external sources can be used to generate a list of cases for audit.

# Chapter 2. Registering a New Algorithm

## 2.1 Introduction

An algorithm is the method or function run by the system in order to produce a random sample of cases. A sample algorithm has been provided which uses a starting point and an interval to determine the list of cases to be included in the case audit.

## 2.2 Creating a New Algorithm

An organization has the ability to add a new algorithm if the sample algorithm is not suitable, ensuring that a different method is used when producing a random sample of cases for audit.

The following example outlines how to add an algorithm 'Every Nth Case', which adds every nth case to the list of cases to be included in the audit. N is a parameter specified by the audit coordinator. Chapter 3, "Utilizing Dynamic Selection Queries," on page 7 describes how to include algorithm parameters in case audit generation. The following sections describe in detail the steps required to create a new algorithm and add it to the application. The steps required are -

- Administratively Define the New Algorithm
- Provide an Implementation for the Algorithm
- Add a Binding to the New Algorithm Implementation

### 2.2.1 Administratively Define the New Algorithm

Add a new custom entry to CT_SamplingStrategy.ctx called Every Nth Case.

```
<code
          default="false"
          java_identifier="EVERYNTHCASE"
          status="ENABLED"
          value="SAMPLEVALUE"
        >
          <locale
            language="en"
            sort_order="0"
          >
            <description>Every Nth Case</description>
            <annotation/>
          </locale>
        </code>
```

*Figure 1. Defining the Algorithm*

### 2.2.2 Provide an Implementation for the Algorithm

The next step that is required to register a new algorithm is to provide the implementation for the algorithm. This implementation must implement the `SamplingStrategy` Interface. The `SamplingStrategy` Interface has one method `getRandomSample`. This method takes a list of case identifiers and applies a sampling strategy to the list to generate a random case sample for audit. It accepts three parameters:

- masterList - the main list of cases that the sample is to be created from
- sampleSize - the number of cases that are to be included in the sample
- properties - a map of algorithm or configuration parameters that can be used in the filtering process.

The `SamplingStrategy` Interface can be found in the package curam.samplingstrategy.impl

```
/*
 * Copyright 2011 Curam Software Ltd.
 * All rights reserved.
 *
 * This software is the confidential and
 * proprietary information of Curam Software, Ltd.
 * ("Confidential Information"). You shall not
 * disclose such Confidential Information and shall
 * use it only in accordance with the terms of the
 * license agreement you entered into with Curam Software.
 */
package curam.samplingstrategy.impl;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import curam.util.exception.AppException;
import curam.util.exception.InformationalException;

public class EveryNthCase implements SamplingStrategy {

  public List<Long> getRandomSample(List<Long> masterList,
    int sampleSize, Map<String, Object> properties)
    throws AppException, InformationalException {

    List<Long> randomSampleList = new ArrayList<Long>();
    Integer n = (Integer) properties.get("n");
    int index = 0;

    if (n <= masterList.size()) {

      while (randomSampleList.size() < sampleSize) {
        if (index + n < masterList.size()) {
          index = index + n;

          // If the element has been returned already,
          // try the next element until one that hasn't
          // been returned is found
          while (randomSampleList.contains(
            masterList.get(index))) {
            if (index < masterList.size() - 1) {
              index++;
            } else {
                index = 0;
            }
          }

          randomSampleList.add(masterList.get(index));
        } else {

          // Run out of elements, loop back to the
          // start of the list
          int elementsToStartOfList = masterList.size() - index;
          index = n - elementsToStartOfList;

          // If the element has been returned already,
          // try the next element until one that hasn't
          // been returned is found
          while (randomSampleList.contains(
            masterList.get(index))) {
            if (index < masterList.size() - 1) {
              index++;
            } else {
                index = 0;
            }
          }

          randomSampleList.add(masterList.get(index));
        }
      }
    }
    return randomSampleList;
```

## 2.2.3 Add a Binding to the New Algorithm Implementation

Guice bindings are used to map the algorithm to the correct algorithm implementation.

```
/*
 * Copyright 2011 Curam Software Ltd.
 * All rights reserved.
 *
 * This software is the confidential and proprietary
 * information of Curam Software, Ltd.
 * ("Confidential Information").  You shall not
 * disclose such Confidential Information and shall
 * use it only in accordance with the terms of the
 * license agreement you entered into with Curam Software.
 */
package curam.samplingstrategy.impl;

import com.google.inject.AbstractModule;
import com.google.inject.multibindings.MapBinder;
import curam.codetable.impl.SAMPLINGSTRATEGYEntry;

/**
 * Guice module for binding Sampling Strategies.
 *
 */
public class Module extends AbstractModule {

  @Override
  public void configure() {

    // register sampling strategies
    final MapBinder<SAMPLINGSTRATEGYEntry, SamplingStrategy>
      samplingStrategies = MapBinder.newMapBinder(binder(),
        SAMPLINGSTRATEGYEntry.class, SamplingStrategy.class);

    samplingStrategies.addBinding(
      SAMPLINGSTRATEGYEntry.EVERYNTHCASE).to(EveryNthCase.class);
  }
}
```

*Figure 3. Binding the Algorithm*

The new algorithm is now ready to be associated with a Case Audit Configuration in the Administration Application. For more information on Case Audit Configuration see the Cúram Case Audits Business Guide.

# Chapter 3. Utilizing Dynamic Selection Queries

## 3.1 What is a Dynamic Selection Query?

Dynamic Selection Queries are used to generate a random sample of cases and contain the selection criteria that are used to search for and produce the list of cases. They allow the audit coordinator to enter any combination of selection criteria to be used when producing a list of cases.

## 3.2 Why use a Dynamic Selection Query?

Four sample queries are provided for each of the standard case types: Integrated Case, Benefit Product Delivery, Liability Product Delivery and Investigation Case. If these queries do not contain sufficient criteria, custom selection queries can be created.

**Important:** Complex queries are not suited to Dynamic Selection Queries as they will not perform. If a complex query needs to be created consider using standard practices followed in the application instead of using a Dynamic Selection Query. The sample queries have been implemented in this way as they are quite complex.

## 3.3 Implementing a Dynamic Selection Query

For new dynamic selection queries a UIM page must be created so that an audit coordinator can enter values for the new selection criteria. Similarly, if a new algorithm requiring parameters is to be used, then this input screen must be developed. See 3.5, “Example: Implementing a Dynamic Selection Query,” on page 8 for an example using the 'Every Nth Case' algorithm. The following outlines the steps required to use a custom Dynamic Selection Query.

- Create a new UIM that contains the required selection criteria. The name of this UIM must match the name that the systems administrator enters for the Random Generation page name when configuring selection queries.
- Any other required input screens must also be developed, such as screens for entering algorithm parameters, entering the number of cases to be returned etc. If multiple screens are used they should be developed as a wizard. For more information on wizards, please consult the `Cúram Web Client Reference Manual`.
- Create the necessary struct to cater for the selection criteria.
- Create and implement a new facade method that is responsible for generating the list of cases for audit. Note that a Case Audit Query Management API is available to help generate this list of cases.
- The systems administrator must create, validate and publish a new Selection Query with the SQL required to retrieve the data and the selection criteria associated with it. The selection query must then be associated with the relevant case audit configuration.

An example of these steps is provided in 3.5, “Example: Implementing a Dynamic Selection Query,” on page 8

## 3.4 The Case Audit Query Management API

The Case Audit Query Management API is a public API used to run selection queries. To use this API a new Dynamic Selection Query must be created in the systems administration application. The Selection Query should contain the SQL that is required to perform the search. For example, if a new search that simply contains a status is required, a dynamic selection query must be entered that contains the selection criteria page name, along with the following SQL -

SELECT caseID INTO :caseID FROM CaseHeader WHERE statusCode = :statusCode

The Case Audit Query Management API contains one main method of interest runDynamicQueryCaseSearch. This method takes two arguments a selection query identifier and a map of selection criteria (as entered by the audit coordinator). The selection query is retrieved and the selection criteria entered by the audit coordinator are substituted into the SQL. The query is then run against the database and a list of CaseHeader records is returned.

```
/**
   * Executes a Case Audit dynamic selection query
   * and returns a list of case header records
   * that match the criteria specified in the
   * dynamic query.
   *
   * @param selectionQueryID the unique identifier of the dynamic
   *         selection query
   *
   * @param parameterMap Map of all parameters
   *        in name value pairs.
   * @return A list of Case Header records that satisfy the
   *         selection criteria
   * @throws AppException
   * @throws InformationalException
   */
    public List<CaseHeader> runDynamicQueryCaseSearch(
    final long selectionQueryID,
    final HashMap<String, String> parameterMap)
    throws AppException, InformationalException
```

*Figure 4. The Case Audit Query Management API*

## 3.5 Example: Implementing a Dynamic Selection Query

Step 1: Create a new UIM that contains the required selection criteria.

This screen allows the audit coordinator to enter selection criteria relating to the dynamic query.

```
<PAGE
  PAGE_ID="exampleSelectionCriteria"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd"
>

  <PAGE_TITLE>
    <CONNECT
      <SOURCE
        NAME="TEXT"
        PROPERTY="PageTitle.Title"
      />
    </CONNECT>
  </PAGE_TITLE>

  <SERVER_INTERFACE
    CLASS="ExampleFacade"
    NAME="ACTION"
    OPERATION="validateCustomCriteria"
    PHASE="ACTION"
  />

  <PAGE_PARAMETER NAME="auditPlanID"/>
  <PAGE_PARAMETER NAME="queryID"/>

  <ACTION_SET
    ALIGNMENT="CENTER"
    TOP="false"
  >
    <ACTION_CONTROL
      LABEL="ActionControl.Label.Cancel"
      ALIGNMENT="LEFT"/>

    <ACTION_CONTROL
      DEFAULT="true"
      IMAGE="NextButton"
      LABEL="ActionControl.Label.Next"
      TYPE="SUBMIT"
    >
      <LINK
        SAVE_LINK="false"
        DISMISS_MODAL="false"
        PAGE_ID="exampleSelectAmount"
      >
        <CONNECT>
          <SOURCE
            NAME="PAGE"
            PROPERTY="auditPlanID"
          />
          <TARGET
            NAME="PAGE"
            PROPERTY="auditPlanID"
          />
        </CONNECT>
        <CONNECT>
          <SOURCE
            NAME="ACTION"
            PROPERTY="result$status"
          />
          <TARGET
            NAME="PAGE"
            PROPERTY="status"
          />
        </CONNECT>
        <CONNECT>
          <SOURCE
            NAME="PAGE"
            PROPERTY="queryID"
          />
          <TARGET
            NAME="PAGE"
            PROPERTY="queryID"
          />
```

Step 2: If required, create a screen to allow the audit coordinator enter the number of cases to audit.

```
<PAGE
  PAGE_ID="exampleSelectAmount"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd"
>

  <PAGE_TITLE>
    <CONNECT>
      <SOURCE
        NAME="TEXT"
        PROPERTY="PageTitle.Title"
      />
    </CONNECT>
  </PAGE_TITLE>

  <SERVER_INTERFACE
    CLASS="ExampleFacade"
    NAME="ACTION"
    OPERATION="validateNumberOfCases"
    PHASE="ACTION"
  />

  <PAGE_PARAMETER NAME="auditPlanID"/>
  <PAGE_PARAMETER NAME="status"/>
  <PAGE_PARAMETER NAME="queryID"/>

  <CLUSTER
    DESCRIPTION="Cluster.Description.Text"
    LABEL_WIDTH="30">

    <FIELD LABEL="Field.Label.Number">
      <CONNECT>
        <TARGET
          NAME="ACTION"
          PROPERTY="key$numberOfCases"
        />
      </CONNECT>
    </FIELD>
  </CLUSTER>


  <ACTION_SET
    TOP="false"
  >
    <ACTION_CONTROL
    LABEL="ActionControl.Label.Cancel"
    ALIGNMENT="LEFT"/>
    <ACTION_CONTROL
      DEFAULT="true"
      IMAGE="NextButton"
      LABEL="ActionControl.Label.Next"
      TYPE="SUBMIT"
    >
      <LINK
        SAVE_LINK="false"
        DISMISS_MODAL="false"
        PAGE_ID="exampleConfigureAlgorithm"
      >
        <CONNECT>
          <SOURCE
            NAME="PAGE"
            PROPERTY="auditPlanID"
          />
          <TARGET
            NAME="PAGE"
            PROPERTY="auditPlanID"
          />
        </CONNECT>
        <CONNECT>
          <SOURCE
            NAME="PAGE"
            PROPERTY="status"
          />
```

If required, create a screen to allow the audit coordinator enter algorithm parameters.

```
<PAGE
  PAGE_ID="exampleConfigureAlgorithm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd"
>

  <PAGE_TITLE>
    <CONNECT>
      <SOURCE
        NAME="TEXT"
        PROPERTY="PageTitle.Title"
      />
    </CONNECT>
  </PAGE_TITLE>

  <SERVER_INTERFACE
    CLASS="ExampleFacade"
    NAME="ACTION"
    OPERATION="generateExampleCaseList"
    PHASE="ACTION"
  />

  <PAGE_PARAMETER NAME="auditPlanID"/>
  <PAGE_PARAMETER NAME="status"/>
  <PAGE_PARAMETER NAME="numberOfCases"/>
  <PAGE_PARAMETER NAME="queryID"/>

  <CONNECT>
    <SOURCE
      NAME="PAGE"
      PROPERTY="auditPlanID"
    />
    <TARGET
      NAME="ACTION"
      PROPERTY="key$auditPlanID"
    />
  </CONNECT>
  <CONNECT>
    <SOURCE
      NAME="PAGE"
      PROPERTY="status"
    />
    <TARGET
      NAME="ACTION"
      PROPERTY="key$status"
    />
  </CONNECT>
  <CONNECT>
    <SOURCE
      NAME="PAGE"
      PROPERTY="numberOfCases"
    />
    <TARGET
      NAME="ACTION"
      PROPERTY="key$numberOfCases"
    />
  </CONNECT>
  <CONNECT>
    <SOURCE
      NAME="PAGE"
      PROPERTY="queryID"
    />
    <TARGET
      NAME="ACTION"
      PROPERTY="key$selectionQueryID"
    />
  </CONNECT>

  <CLUSTER LABEL_WIDTH="30">
    <FIELD LABEL="Field.Label.Interval">
      <CONNECT>
        <TARGET
          NAME="ACTION"
```

Step 3: Create the necessary struct to cater for the selection criteria.

This struct contains all selection criteria available for the selection query along with any other required parameters.

ExampleSelectionCriteria

- long auditPlanID
- long selectionQueryID
- int numberOfCases
- int interval
- String status

Step 4: Create and implement a new façade method that is responsible for generating the list of cases for audit.

This method uses the Case Audit Query Management API to execute the dynamic query, using the supplied selection criteria. This returns the list of cases matching the selection criteria. The relevant algorithm is then invoked to filter the case list. Finally, case audit records are created for each case in the remaining list.

```
// Inject the map
  @Inject
  private Map<SAMPLINGSTRATEGYEntry, SamplingStrategy>
    samplingStrategies;

  /**
   * Generates the sample list of cases for audit based on the
   * supplied selection criteria. This method filters the list
   * using the algorithm associated with the case type for this
   * audit plan. The number of cases returned in the list is also
   * restricted by the number of cases specified by the user.
   *
   * @param key The selection criteria, selection query
   *            identifier, the audit plan identifier,
   *            the number of cases to generate and any
   *            algorithm parameters.
   *
   * @throws AppException
   * @throws InformationalException
   */
  public void generateExampleCaseList(
    ExampleSelectionCriteria key)
    throws AppException, InformationalException {

    AuditPlan auditPlan = auditPlanDAO.get(key.auditPlanID);

    CaseAuditQueryManagement caseAuditQueryManagement =
      new CaseAuditQueryManagement();

    // Add all selection criteria to the map
    HashMap<String, String> parameterMap =
      new HashMap<String, String>();
    parameterMap.put(":statusCode", key.status);

    // Call the Case Audit Query Management API
    // to run the selection query
    List<curam.piwrapper.caseheader.impl.CaseHeader> caseList =
      caseAuditQueryManagement.runDynamicQueryCaseSearch(
        key.selectionQueryID, parameterMap);

    // Get the algorithm/sampling strategy configured
    // for this case type
    final SamplingStrategy samplingStrategy =
      samplingStrategies.get(
        auditPlan.getAuditCaseConfig().getAuditAlgorithm());

    List<Long> caseIDList = new ArrayList<Long>();

    for (CaseHeader caseHeader : caseList) {
      caseIDList.add(caseHeader.getID());
    }

    // Set up the algorithm parameters
    Map<String, Object> params = new TreeMap<String, Object>();
    params.put("n", new Integer(key.interval));

    // Invoke algorithm to generate case sample,
    // passing in the list of cases,
    // the number of cases to return and the algorithm parameters
    List<Long> caseIDs = samplingStrategy.getRandomSample(
      caseIDList, key.numberOfCases, params);

    curam.core.facade.intf.CaseAudit caseAuditObj =
      curam.core.facade.fact.CaseAuditFactory.newInstance();

    // for each case, create a case audit
    for (int i = 0; i < caseIDs.size(); i++) {

      CaseAuditDetails caseAuditDetails = new CaseAuditDetails();
      caseAuditDetails.dtls.auditPlanID = key.auditPlanID;
      caseAuditDetails.dtls.caseID = caseIDs.get(i);
      caseAuditObj.createCaseAudit(caseAuditDetails);
    }
```

Step 5: The systems administrator must create, validate and publish a new Selection Query with the SQL required to retrieve the relevant data and the selection criteria associated with it.

**Important:** Appropriate database indexing should be provided for any custom dynamic selection queries. Also, if a significant case load is expected to be returned from the selection query, it would be advisable to consider using Deferred Processing to generate the random sample of cases. For more information on Deferred Processing, please see the Cúram Server Developer Guide.

## 3.6 Using Dynamic Selection Queries for Manual Case Selection

Dynamic selection queries can also be used for manual case selection. A similar process to the one described above can be used. A new UIM must be created that contains the required selection criteria. Note, this new UIM must be created because the resulting case list must be included in the page. This allows the audit coordinator to manually select from the list of cases. The name of this UIM must match the name that the systems administrator entered for the Manual Search page name in the Selection Query configuration. The Case Audit Query Management API can again be used to execute the query.

# Chapter 4. Configuring Selection Queries

## 4.1 Introduction

This chapter describes the configuration options available for Selection Queries. Selection Queries are used to generate a sample of cases and contain the selection criteria that are used to search for and produce the list of cases. Two types of selection query exist, dynamic queries and fixed queries.

## 4.2 Dynamic Selection Queries

A dynamic selection query, when configured for audit, presents a page containing selection criteria. An audit coordinator must enter values for the criteria which will return a case sample. The audit coordinator can simply enter the parameters for one criterion, or has the flexibility to enter parameters for any logical combination of parameters. For example, to return all open cases for males starting 1-6 January, the audit coordinator would enter values as follows: case status of open; case start date range of 1-6 January; and gender of male.

## 4.3 Fixed Selection Queries

A fixed selection query provides a predefined set of selection criteria that is defined through the entry of an SQL statement. The fixed selection query when created by a systems administrator contains the values for the selection criteria such as case status of open etc. An audit coordinator has the option to select this fixed query when creating a case sample, however, no audit coordinator entry of selection criteria is required as they have already been input as part of the query.

## 4.4 Creating a Selection Query

Creating a selection query is a two step process; the first step allows the addition of the basic selection query details, such as name and the query type, along with the SQL. The second step allows the entry of the selection criteria associated with the selection query.

The 'Name' of the selection query is what will be displayed to the administrator when configuring a case audit and to the audit coordinator when generating a random case sample, so it should have a meaningful and descriptive name.

'Query' represents the type of objects that the selection query will impact. In this initial release there is one type of object, Case. Additional functionality is envisaged in this area so that selection queries can be captured for any object. An example usage of this might be participants, where an agency may want to poll all employers to determine information on employee working patterns.

The 'Query Type' should be chosen depending on the type of selection query required, Dynamic or Fixed. If the selection query is 'Dynamic', the 'Random Generation' and 'Manual Search' page names must be entered. The page names entered must match the name of the custom UIM that will be used to enter the selection criteria.

The SQL text is the SQL statement that will be used to execute the selection query. Every field on the custom selection criteria selection screen should be part of the WHERE clause. Take the example outlined in Chapter 3, “Utilizing Dynamic Selection Queries,” on page 7 a UIM has been created to allow the audit coordinator to select a status, the SQL necessary for this query would be -

```
SELECT caseID INTO :caseID FROM CaseHeader WHERE statusCode = :statusCode
```

### 4.4.1 Adding Selection Criteria

The next step is to enter the attribute names and values to be used in the SQL query. These can be added using the 'Add Criteria' link on the second page of the wizard.

For dynamic selection queries the values will only be used for validation purposes. Validation of a selection query will be discussed in the next section. For fixed selection queries the values entered will be the actual values used when executing the query as the audit coordinator will not have the option to enter their own values. 'Name' should contain the attribute as it appears in the WHERE clause of the SQL statement. The 'Display Name' and 'Display Value' should be acceptable representations of 'Name' and 'Value' that can be displayed to an audit coordinator.

## 4.5 Validating a Selection Query

Once a selection query has been created it can then be validated. The validation performed checks that the SQL query is valid and that the statement is only attempting to read data. Note, database integrity must be maintained so the SQL statement should not modify or remove data. To minimize this risk, the SELECT and INTO clauses are defaulted. As validation errors can be quite complex please ensure that the SQL query provided adheres to these guidelines.

## 4.6 Publishing a Selection Query

A selection query must be published to make it available to administrators to associate with a case audit configuration. This ensures that a query is validated before being made available for use to an administrator. Finally, before the selection query can be used by an audit coordinator as part of an audit plan, it will need to be associated to a case audit configuration by the administrator.

# Chapter 5. Case Audits Web Service

## 5.1 Case Audits Web Service

A Case Audits Web Service has been provided to allow case data from external sources to be audited. The Web Service can be invoked on by any capable Web Service client, including tools such as the Business Intelligence and Reporting Tools (BIRT) reporting tool. The Web Service receives a list of case identifiers to be used in an audit along with an associated name to identify the data set, from an external source. This data is stored in the ExternalCaseAuditData and ExternalCaseAuditDataItem tables, ready to be included in an audit plan. The audit coordinator will have the option to select case data from this source. If any of the cases do not exist on the system, no data will be saved and a response indicating an error has occurred will be returned. The Case Audit Web Service can be found at curam.core.ws.convert.bs.impl.ExternalCaseAuditData

# 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.

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

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

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 is for planning purposes only. The information herein is subject to change before the products described become available.

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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. 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 http://www.ibm.com/legal/us/en/copytrade.shtml.

BIRT is a registered trademark of Eclipse Foundation.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.

Printed in USA