IBM WebSphere Business Integration Adapters

Adapter Development Kit User Guide

Version 2.4.0

Add Custom Cover Content Here

IBM WebSphere Business Integration Adapters

Adapter Development Kit User Guide

Version 2.4.0

Note!

Before using this information and the product it supports, read the information in "Notices," on page 11.

19December2003

This edition of this document applies to IBM WebSphere Business Integration Adapter Development Kit version 2.4.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about IBM WebSphere Business Integration documentation, e-mail doc-comments@us.ibm.com. We look forward to hearing from you.

Contents

About this document

The IBM^{R} WebSphere^{R} Business Integration Adapter portfolio supplies integration connectivity for leading e-business technologies, enterprise applications, and legacy and mainframe systems. The product set includes tools and templates for customizing, creating, and managing components for business process integration.

This document describes how to configure and run samples for the adapter development kit (ADK) that are included in the IBM WebSphere Business Integration Adapters product.

Audience

This document is for consultants, developers, and system administrators who use the adapter development kit at customer sites.

Related documents

The complete set of documentation available with this product describes the features and components common to all WebSphere Business Integration Adapters installations, and includes reference material on specific components.

The complete set of documentation available with this product describes the features and components common to all WebSphere Business Integration Adapters installations, and includes reference material on specific components.

You can install related documentation from the following sites:

- For general adapter information; for using adapters with WebSphere message brokers (WebSphere MQ Integrator, WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker); and for using adapters with WebSphere Application Server, see the IBM WebSphere Business Integration Adapters InfoCenter:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- For using adapters with WebSphere InterChange Server, see the IBM WebSphere InterChange Server InfoCenters:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For more information about WebSphere message brokers:
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- For more information about WebSphere Application Server:
  http://www.ibm.com/software/webservers/appserv/library.html

These sites contain simple directions for downloading, installing, and viewing the documentation.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.

italic	Indicates a new term the first time that it appears, a variable name, or a cross-reference.
blue text	Blue text, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click any blue text to jump to the object of the reference.
`{ }`	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
|	In a syntax line, a pipe separates a set of options from which you must choose one and only one.
`[ ]`	In a syntax line, square brackets surround an optional parameter.
`...`	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.
`< >`	Angle brackets surround individual elements of a name to distinguish them from each other, as in `<server_name><`*`connector_name`*`>tmp.log`.
`/, \`	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes.
`%text%` and `$`*`text`*	Text within percent (%) signs indicates the value of the Windows `text` system variable or user variable. The equivalent notation in a UNIX environment is `$`*`text`*, indicating the value of the *`text`* UNIX environment variable.

New in this release

The Adapter Development Kit User Guide is new in the release of WebSphere Business Integration Adapter Framework version 2.4.

Chapter 1. TwineBall

The TwineBall sample consists of a connector, ODA and application, designed to demonstrate some basic principles of adapter development. The application is called TwineBall, and it provides:

- An RMI interface for basic data-oriented operations such as create, retrieve, update, and delete
- Entity definition information to the Object Discovery Agent, which is included

Audience

This document is geared toward seasoned Java developers looking at developing custom adapters. Before you begin to develop an adapter, you should also study the *IBM WebSphere Connector Development Guide for Java* and the *IBM WebSphere Business Object Development Guide* in the documentation Infocenter.

For a general overview of adapters, refer to the *Technical Introduction to WebSphere Business Integration Adapters* in the documentation Infocenter.

Adapter architecture

The TwineBall server uses a database created in DB2 to store data. This server can handle many different entity definitions, but the tables most often used are:

- Customer
- Address
- Order
- OrderLine

Figure 1 on page 2 shows the TwineBall architecture.

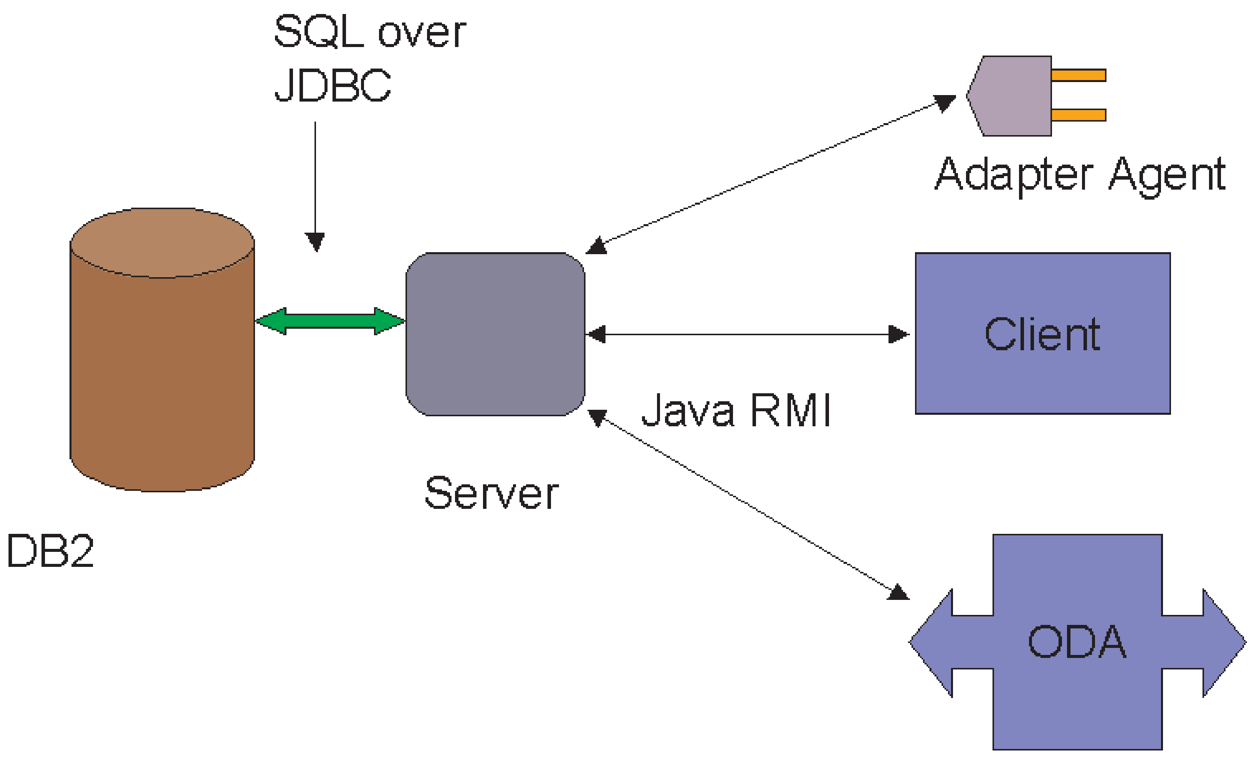

Figure 1. Original TwineBall architecture

Event delivery is provided by a polling mechanism. For polling to work, the user must create event and archive tables in DB2, and triggers must also be in place.

The adapter and ODA each connect to this server via Java RMI. The adapter and ODA each take a server URL that represents the server name, port, and service name.

The standard server URL is `RMI://localhost:2112/TwineBall`.

Alternately, the TwineBall adapter can run independent of the server. In this case it calls an alternate implementation of the TwineBall API, written in C++. The adapter will load a C++ shared library and call into that instead of making remote RMI calls. For more information refer to "Using JNI" on page 5.

The TwineBall server

The TwineBall server components are listed below.

- **TwineBallConnection**: Contains most of the database logic.
- **TwineBallServer**: Provides connections to RMI clients.
- **TwineBallInterface**: Provides an API for the Adapter, ODA, or other client to call.
- **TwineBallConnectionPool**: Pools TwineBall Connections (considered part of the API).

The adapter

The adapter components are listed below.

- **AdapterAgent**: Extends CWConnectorAgent by providing initialization and termination logic.
- **AdapterBOHandler** : Extends CWConnectorBOHandler: Handles service call requests.
- **AdapterEventStore**: Extends CWConnectorEventStore by providing event delivery logic.

The Object Discovery Agent

The ODA components are listed below.

- **ObjectDiscoveryAgent** : Extends ODKAgentBase by providing initialization, and calls ObjectFinder and ObjectAnalyzer.
- **ObjectFinder** : Implements the logic for getTreeNodes.
- **ObjectAnalyzer** : Implements the logic for generateDefs.

Installed directories

Several directories are installed with the TwineBall sample. They are listed below.

Table 1. File directories

Directory	**Purpose**
/dependencies	Properties and policy files required to run the TwineBall Server.
/documentation	Contains a README file on how to use the sample.
/JNI_supplement	Contains code that demonstrates how to call a C++ API in an adapter using JNI.
/Repository	Contains the message files for the adapter and ODA.
/Samples	Contains business objects for the adapter.
/setup	Contains scripts for setup, as well as running the server, adapter, and Object Discovery Agent.

Java source package

The package that contains the Java resources is structured as follows.

Table 2. Java resources

Resource	**Contents**
com.ibm.wbia.TwineBall.Adapter	The TwineBall adapter
com.ibm.wbia.TwineBall.oda	The TwineBall Object Discovery Agent
com.ibm.wbia.TwineBall.Server	The TwineBall server and API
com.ibm.wbia.TwineBall.jnibridge	The JNI supplement (stub library and vector bridge)

Getting up and running

The TwineBall server, adapter, and ODA have been tested for Windows only. They may work on other platforms, but testing them is left as an exercise for the user.

To run the TwineBall sample, you must have a JDK installed (preferably 1.3.1).

Setting up the database

In order to demonstrate create, restore, update, and delete (CRUD) operations and true metadata-driven design, the TwineBall server uses an actual database to define its entities and store its data. This necessitates some additional setup. Scripts have been provided to make this easier, but these scripts will require some slight modification to match your system.

To set up the database, follow these steps.

1. Install the WebSphere Business Integration Adapter Framework or WebSphere InterChange Server and all their pre-requisites.
2. Install IBM DB2 Version 8, Fix pack 2.
3. Create the following user ids in your operating system:
   TwineBall, TwineBallAdapter.
   Give them the password
   ```
   sample42
   ```
4. Create the `Twine` database. In the `setup/SQL` directory, you'll find Twine-database.bat and Twine-database.sql.
   Edit Twine-database.sql to match your password for the `db2admin` id.
5. Run the Twine-database.bat script. You should not receive any error messages. If you do, resolve them before continuing.
6. Create the tables needed by the TwineBall server. In the `setup/SQL` directory, you'll find TwineBall-Schema.bat and TwineBall-Schema.SQL.
7. Run the TwineBall-Schema.bat script. Resolve any errors before continuing.
8. Set up the Event and Archive tables. In the `setup/SQL` directory, you'll find WBIA-Events.sql and WBIA-Events.bat.
9. Run the WBIA-Events.bat script. Resolve any errors before continuing.

Running the server

To run the server, follow these steps.

1. Compile the entire Java source tree in both the `/src` and `JNI_Supplement` directories by importing all the Java source into a project in an IDE. Configure the IDE to have all the JAR files in the `/lib`directory of the InterChange Server or Adapter Framework in the CLASSPATH.
2. Create a directory called `TwineBall` under the `connectors` directory of your Adapter Framework or InterChange Server installation.
3. Export all compiled class files into this directory using a normal Java package structure (that is, `com.ibm.sample.MyClass` stored as `com/ibm/sample/Myclass.class`).
4. Copy all the files in the `setup\scripts` directory to this folder as well. Normally an application would be in a different directory than an adapter, but structuring it this way will simplify the install.
5. Add the newly created `TwineBall` directory to your system CLASSPATH. This will allow the Java runtime to find the server classes.
6. Make sure the Java Development Kit `bin` directory is in the CLASSPATH. Run the runrmic.bat file. This will produce the necessary stubs and skeletons necessary for RMI communication.
7. Create a new directory called `Properties` inside the `TwineBall` directory.
8. Place the TwineBall.policy and TwineBall.db2.properties files from the `dependencies` directory here and rename TwineBall.db2.properties to TwineBall.properties.

9. Edit the TwineBall.properties file to change `server.name` to your system name. For instance, if your machine is called JEFFB, your `server.name` will be `//JEFFB:2112/TwineBallServer`.
10. Start the TwineBall server by executing the `runserver` script. The message "Server ... has been registered" appears when it is working.
11. Leave the server running in a window. Your ODA and adapter will connect to it via RMI.

Starting the TwineBall connector

To start the TwineBall connector, follow these steps.

1. Create a directory called `TwineBall` inside the `/connectors` directory of your Adapter Framework or InterChange Server installation.
2. Put the configuration file in that directory if you are not running the Adapter Framework for WebSphere InterChange Server.
   If you are running InterChange Server, import the configuration.
3. Create the necessary WebSphere MQ queues for running the adapter for your broker. If the broker is InterChange Server, this may not be necessary because you can use the IDL transport.
4. Start up the Connector Configurator and change the `MessageFileName` standard property to "TwineBallAdapter.txt".
5. Attempt to start the connector using the ordinary syntax for starting an adapter. For example:
   ```
   start_TwineBall TwineBall WMQI  -c
   C:\IBM\WebSphereAdapters\connectors\TwineBall\TwineBall.cfg
   ```
6. To test the adapter, you can set up a pass-through collaboration in InterChange Server or reverse the delivery and service call queues in non-InterChange server mode. However, these procedures are beyond the scope of this document.

Starting the TwineBall ODA

To start the TwineBall ODA, follow these steps.

- Attempt to start the ODA by running the start_twineballODA.bat file.
- Attempt to use the ODA by discovering it through Business Object Designer.
- When you select the nodes from which business objects will be created, please select all the business objects you want to create at the same time. Child business objects will not be created unless you specifically select them.

Using JNI

You can build a Java adapter and use the Java Native Interface (JNI) to call the C++ API. Refer to Chapter 2, "The Java Native Interface (JNI)," on page 7 for details.

Chapter 2. The Java Native Interface (JNI)

This chapter describes how to use the Java Native Interface (JNI) with a Java adapter for integration with a C++ application.

There are three basic approaches to enabling an application with a C++ API.

- Build a C++ adapter using the CDK.
- Build a Java adapter and use the JNI (Java Native Interface) to call the C++ API.
- Don't build an adapter at all, but use a technology adapter, such as the adapters for MQSeries or COM.

We will discuss the second option, and look at a sample adapter that utilizes this approach.

Note: This example is intended to show how a Java adapter using JNI might be built, and to show how the sample JNI adapter works. Knowledge of JNI implementation techniques is a prerequisite to understanding and using this approach.

JNI is designed to allow Java applications to call C/C++ libraries. The following example uses Java to call a C++ API. The API is implemented to show how TwineBall, which is a Java adapter, could call a C++ API.

Interaction with the API

The TwineBall adapter makes all its calls to the application server through the TwineBall interface. In Java RMI, calls to a remote interface are automatically marshaled by the JVM. Figure 2 shows the original interaction over Java RMI.

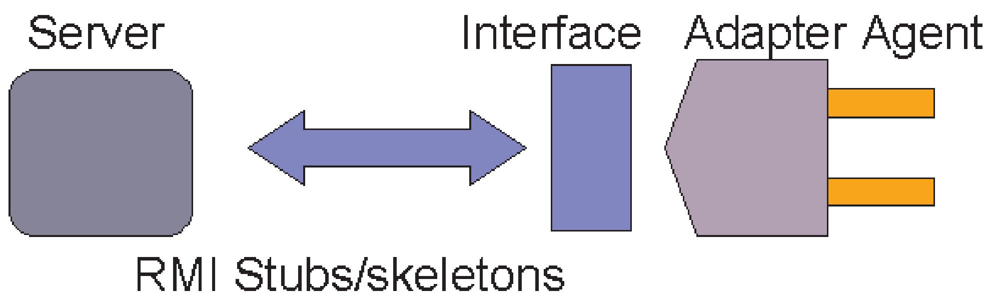

Figure 2. Interaction over RMI

To demonstrate JNI, the server side of the picture has been replaced with a C++ shared library. Now the picture looks like Figure 3 on page 8.

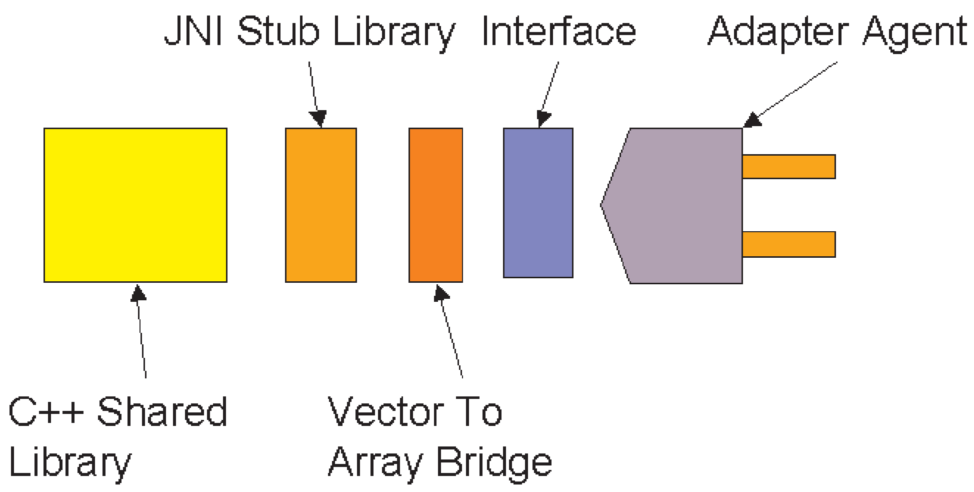

Figure 3. Modified TwineBall architecture

Because a well-defined Java interface was already calling application, very few code changes were required on the adapter side to change the implementation of that interface. This allows us to demonstrate an adapter using JNI and an adapter using Java RMI with the same adapter code. Only the implementation behind the interface changes.

The TwineBall interface makes heavy use of Java vectors. Rather than try to bring vectors into the C++ world, we decided to convert the vectors into arrays of strings. This is what the VectorToArrayBridge does. If it is more convenient for the adapter to use Java utility classes, you may wish to use a similar approach.

The JNI Stub Library declares the native methods. This is what connects the methods implemented in C++ with the Java world.

The C++ shared library in this case is implemented with a single C++ module. However, this shared library includes the JNI.H header file, and is therefore aware of Java types. Conversion between basic Java types and C++ types still largely occurs in the C++ space. Interfacing with a pre-existing C++ application/API will require new C++ code to be written as well as Java.

Please note that a database is no longer involved. In this example, the interface is implemented with pre-determined return values.

In the sample JNI source code, you will find the following pieces.

Table 3. Components of JNI source code

Filename	Function
Makefile.PLATFORM	Builds the C++ shared library on the given platform.
com_ibm_wbia_TwineBall_jnibridge_JNIStubLibrary.cc	Source code for C++ shared library. Contains the C++ “glue code”.
com_ibm_wbia_TwineBall_jnibridge_JNIStubLibrary.h	Header file for C++ shared library.
com.ibm.wbia.TwineBall.jnibridge.JNIStubLibrary	The JNI Stub Library
com.ibm.wbia.TwineBall.jnibridge.VectorToArrayBridge	The Vector to Array Bridge

Table 3. Components of JNI source code (continued)

Filename	Function
com.ibm.wbia.TwineBall.jnibridge.test.TestDriver	Test driver for JNIStubLibrary

Compiling and running the TwineBall JNI implementation

We use Make to build the shared library. Ensure you have a complete build environment for the target platform. Each platform's makefile contains the information about which compiler it is built for. If that compiler and Make are installed, all you should have to do to compile the library is to extract the files onto the target platform and type `make –f Makefile.Platform` where `Platform` is the desired target platform.

To run the TwineBall adapter with the JNI bridge:

1. Install either WebSphere InterChange Server or the WebSphere Business Integration Adapter Framework on the target platform.
2. Put the compiled classes for the TwineBall server and the adapter in the `%CROSSWORLDS%/connectors/twineball` directory.
3. Put the message file from "dependencies" in the "messages" folder.
4. Put the shared library and the `start_twineball.sh` script in the `%CROSSWORLDS%/connectors/twineball` directory.
   You will not need to set up the TwineBall database or server because in this example, the adapter will call the JNI bridge instead.
5. Configure the adapter, using twineball.cfg as a template, and use the special string `JNIStub` as the server URL.
6. Start the adapter normally using its start script.

The JNI backend for the adapter implements both service call and event delivery for "TwineBallCustomer" business objects. You will find the TwineBallCustomer business object in the "sample business objects" folder.

Building a JNI adapter

To build a JNI adapter, follow these steps.

1. Examine the existing C++ API and note any data structures that are not defined in Java. You will need to convert these in your C++ glue code.
2. Create a JNI Stub library with native declarations that match those methods as closely as possible in Java. You will need to do this for every function exposed in the API that the adapter will use.
3. Run `javah` to create a C++ header file for the stub library. Example usage:
   `javah com.ibm.wbia.TwineBall.jnibridge.JNIStubLibrary`
4. Implement every function in the resulting header in C/C++. You will use this code to glue the JNI function signatures to your API.
5. Test every function in the JNI Stub library with a test driver program.
6. Begin coding the adapter as you would a Java adapter, using the JNI Stub library as the Java API.

Notes:

1. As we saw with TwineBall, it may be appropriate to do type conversions on both sides of the JNI layer.

2. Character sets are very important. When we convert Java Unicode strings to C++ strings in this example, we convert them to UTF-8. If your application does not handle UTF-8, you'll need to convert them to your particular codepage. Be very careful when you do this to either handle all possible Unicode characters, or fail appropriately.
3. On many versions of HP-UX, you will need to declare and call a `_main()` function in a shared library to do JNI with C++. This tells the operating system to initialize the C++ runtime. The same code on Win32 will fail, so `#defines` may be necessary for this porting issue.

Thread safety

Note these points about threading.

- Remember that Java adapters are multi-threaded by default. Your adapter could be asked to handle multiple service calls on multiple threads simultaneously. If the native code is not threadsafe, you may need to single-thread the adapter by using the `-t` option in the start script.
- Do not pass Java objects from one thread to another in C++ space, though it's unlikely you'd need to do that in an adapter anyway.
- Normal rules for synchronization of globally accessible objects apply.
- You must not pass `JNIEnv*` from one thread to another, or depend on the pointer being the same in two different threads.
  The JVM creates a distinct JNI environment for each thread, and the native code must use only the pointer valid for the current thread.

Appendix. Notices

IBM may not offer the products, services, or features discussed in this document in all 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 give 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.

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 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 RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195

Raleigh, NC 27709-2195
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 measurement 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 necessarily 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.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may 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.

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

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM WebSphere Business Integration Adapter Framework V2.4.0.

Printed in USA