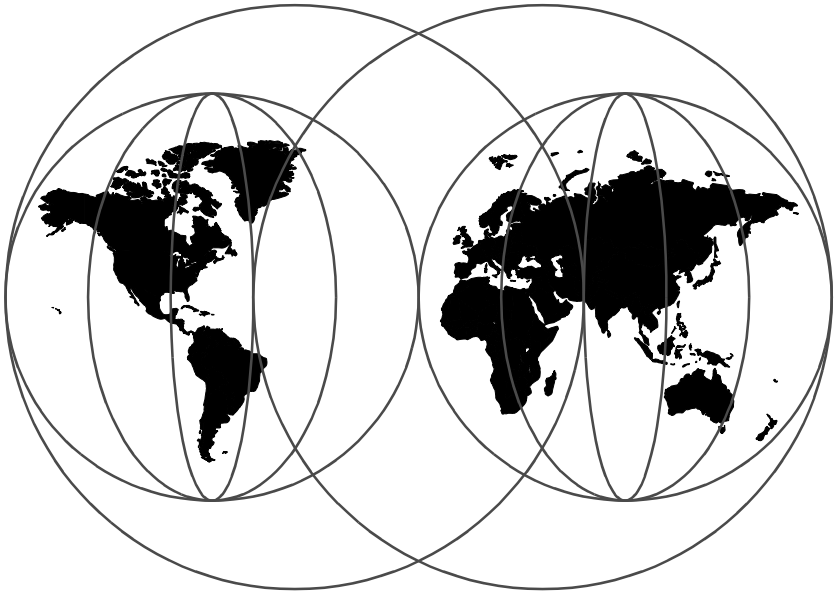
# IBM CBConnector Cookbook Collection: CBConnector Bank User Guide

*Alex Gregor, Henri Jubin, The Team*



**International Technical Support Organization**

http://www.redbooks.ibm.com

SG24-5121-00

SG24-5121-00

International Technical Support Organization

# IBM CBConnector Cookbook Collection
# CBConnector Bank User Guide

August 1998

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 63.

**First Edition (August 1998)**

This edition applies to Component Broker Connector Version 1, Release 2 for use with Windows NT/4 Service Pack 3.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

# Contents

# Figures

# Tables

# Preface

This redbook is a user guide for installing and running the CBConnector Bank, which is a sample bank application developed for Component Broker Connector Release 1.2. The redbook is part of the International Technical Support Organization's (ITSO) *IBM CBConnector Cookbook Collection*.

Component Broker Connector (CBConnector) is a new member of the IBM Transaction Series product family that supports distributed object computing in a multitier environment. CBConnector provides a middle-tier application that allows Business Objects to be highly managed and integrated with back-end databases and transactional systems. Different types of first-tier clients can access the middle-tier Business Objects. The middle tier essentially includes middleware that provides clients with an object-oriented rendering of other middleware. In this regard, CBConnector is not a stand-alone product, but instead, is designed to work with existing resource managers that provide persistence, concurrency control, and other services needed in commercial computing environments.

CBConnector is composed of an industry-leading set of technologies that facilitate distributed object applications. As the only solution of its kind on the market today, it combines three critical dimensions:

- Runtime
- Systems Management
- Development

Component Broker consists of two parts that support these three dimensions. CBConnector provides the runtime environment and supports systems management. The Component Broker Toolkit (CBToolkit) contains the tools that application developers use to define and implement the objects running on the middle tier.

CBConnector's unique value lies in its ability to integrate and therefore leverage existing enterprise transactions and data. It will most often be used to extend existing business models.

We developed the CBConnector Bank as an application that demonstrates the capabilities of Component Broker Release 1.2. This application will grow to encompass new functionality of subsequent releases of Component Broker.

The application has three tiers: the client tier, the middle tier–which is the Component Broker server–and the legacy tier, where a CICS application resides. We call this the "legacy system" in quotes because it is a newly developed system, one whose sole purpose is to act as a persistent store to the bank application. However, this tier would normally be an existing application to be integrated in the new, Object Oriented architecture that Component Broker represents.

The CBConnector Bank makes it possible to quickly set up a complete application at a customer site that shows how an end-user request at a client tier is handled by the Component Broker server at the middle tier, and passed to a legacy system on the legacy tier.

This redbook lists all the hardware and software requirements that have to be met to run the CBConnector Bank application. Then, it provides instructions on how to set up the Component Broker environment and install the CBConnector Bank with its three tiers. Finally, an application scenario is provided that presents an overview of the application functionality and architecture and suggests a banking scenario for an application demo.

Developers can use this book as a supplement to the *CBConnector Bank Implementation* redbook, so that they have a live example of the concepts described in the implementation book.

Sales support personnel can use this book to install and demonstrate a sample application at a customer site, with assistance from technical support professionals during the installation phase.

**Please Note**

The ITSO plans to make the CBConnector Bank CD-ROM available for CBConnector 1.3 during November 1998. The zip file will be downloadable from the following FTP site:

`ftp://www.redbooks.ibm.com/redbooks/SG245121.`

Alternatively, you can go to:

`http://www.redbooks.ibm.com`

and navigate through the Additional Materials tab.

## How This Document is Organized

The chapters follow chronologically the steps needed to be performed to install the Component Broker Connector sample bank application. We introduce each of them by providing a short description of their contents:

- Part 1, "Ready"
  - Chapter 1, “Introduction” on page 3, introduces the CBConnector Bank and its scenarios on the client and server side.
  - Chapter 2, “Hardware and Software Prerequisites” on page 7, specifies the hardware and software requirements for CBConnector and the CBConnector Bank application.
  - Chapter 3, “Installing the Client System and CBConnector Server” on page 15, provides a roadmap for how to install the Component Broker server and how to prepare the CBConnector Bank client system.
  - Chapter 4, “MVS Legacy System Roadmap” on page 17, gives instructions on how to set up your MVS environment (CICS, DB2 and COBOL) for the installation of the "legacy system".
  - Chapter 5, “Replicating the Legacy System to Windows NT” on page 27, provides you with guidelines for porting the MVS system to run on an NT server.
  - Chapter 6, “Installing CBConnector Bank Server Applications” on page 33, gives instructions on how install and configure the CBConnector Bank server applications.
  - Chapter 7, “Installing the CBConnector Bank Client Application” on page 39, gives instructions on how to set up your NT environment to support CBConnector.
- Part 2, "Go"
  - Chapter 8, “Application Scenario” on page 43, describes the CBConnector Bank, gives an introduction to the application architecture, and suggests how to run an application demo.
- Appendixes
  - Appendix A, “Test the Component Broker Connector Installation” on page 55, provides a recipe on how to install two test applications that test your distributed CBConnector development environment.

## Other Redbooks in the CBConnector Series

So far, there are four books in the CBConnector redbook series. Below is a short description of what each book covers.

- *IBM Component Broker Overview,* SG24-2022. This redbook provides a general understanding of CBConnector and examines the underlying architecture and concepts.
- *IBM CBConnector Cookbook Collection* is a series of three redbooks, where the objective is to give the reader hands-on experience by installing and running sample applications and providing step-by-step (or cookbook) instructions on how to develop CBConnector applications. The different parts that make up the Cookbook Collection are:
  - *First Steps,* SG24-2033. This redbook is the introductory book, that leads you through simple examples that show the basic functionality of CBConnector.
  - *CBConnector Bank Implementation,* SG24-5119. This redbook goes one step further, and explains the development process of a sample bank application, from object model to installed Business Objects. It discusses CBConnector specific issues and concerns encountered during this development.
  - *CBConnector Bank User Guide,* SG24-5121. This is the redbook you are reading now, and it provides instructions on how to install and run the bank application discussed in the *CBConnector Bank Implementation* redbook.

In addition to the CBConnector redbooks, we recommend reading the books in the *CBConnector Library*, which is a collection of complete installation guides, programming guides and references.

## The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working for the International Technical Support Organization, Austin Center. The project was designed and managed by Alex Gregor and Henri Jubi. The writers were:

**Dr. Andy Bond** is a Principal Research Scientist at the Co-operative Research Centre for Distributed Systems Technology in Brisbane, Australia. He has over 10 years experience in the field of distributed systems, including load sharing, distributed system architecture design, DCE, CORBA, and experimental middleware technologies. He has written many publications in

these areas. He holds a Ph.D. degree from Victoria University of Wellington, New Zealand, where his topic was "Adaptive Load Sharing in a Distributed Workstation Environment".

**Ryan Cox** is an Object Technology/Application Development specialist working in IBM Advanced Technical Support. He has worked on projects in many areas, including Internet/intranet development, Lotus Notes/Domino, Java, distributed computing, and CORBA. He currently supports application development tools in the IBM VisualAge family, including VisualAge for Java and Component Broker Connector in the IBM Transaction Series.

**Mark Fitzpatrick** is a Principal Consultant at the Distributed Systems Technology Centre in Brisbane, Australia. He has worked in the IT business for 20 years. His areas of expertise include object-oriented technologies, distributed architectures, such as CORBA, and application development in languages such as Smalltalk and Java.

**Alex Gregor** is an IBM Senior Software Engineer working at the International Technical Support Organization (ITSO), Austin Center in the OO/AD group. His responsibilities include technical support for IBM application frameworks.

**Henri Jubin** currently works for the ITSO in Austin, where he covers the area of object-oriented technology and, in particular, the JavaBeans and Java Enterprise arena. Henri has previously worked in various support and consulting positions with IBM France. He has dealt with topics such as object-oriented technology, OS/2, Windows NT, and OpenDoc.

**Zoran Lerch** is a self-employed data-processing consultant and an IBM BestTeam partner, focusing on Java and JavaBeans technologies and on the broad spectrum of their application.

**Dr. Andry Rakotonirainy** is a Senior Research Scientist at the Co-operative Centre for Distributed System Technology Center (DSTC) in Brisbane, Australia. He holds a Ph.D from INRIA (Institut National de Recherche Informatique et Automatique) France for his doctoral work titled "Advanced Transaction Models". He has significant experience in, and has written publications on, distributed systems.

**Hanne Rygg Johnsen** is a Systems Engineer in the Nordic Object Technology Practice (OTP), IBM Norway. She has four years experience in developing object-oriented systems with Smalltalk and Java, and most of them have involved interfacing to legacy systems. As a member of the OTP, she is currently dedicated to the reuse of application frameworks and to

providing customer consulting services in object-oriented analysis and design.

**Pasi Salminen** is a project manager at Profit Ltd. in Finland. He has six years experience in Object Technology and three years in distributed computing. Profit Ltd. is building large-scale insurance systems using Object Technology. Its main product, Once&Done, is a system for managing the complete process from point of sale to contract administration. Profit's goal is to provide insurance companies a path from the current "system jungle" to a component based system architecture conforming to industry standards and running on all major platforms.

**Terje Storstein** has worked as a Systems Engineer in IBM Norway for almost 30 years. He started in the mainframe area, worked with the distributed systems of the 1970s, and went on to work in the client/server era of the 1980s and 1990s. Currently, he is working in the e-business department of the Norwegian Global Services.

**Hennie van Wijk** is a Technology Consultant at Amalgamated Banks of South Africa (ABSA Bank). He is part of a team responsible for defining Application Architecture for ABSA, specifically dealing with Object Technology Strategies and OO Analysis and Design. He has extensive mainframe experience including application development with COBOL, CICS, IMS, and DB2 database analysis and design, which have kept him occupied for the past 12 years. He is currently involved in research and development into retrofitting "legacy systems" for reuse in a distributed computing environment.

Thanks to the following people for their invaluable contributions to this project:

**IBM Development Laboratory in Austin, Texas:**

Mei-Mei Fu
Greg Truty

**IBM Development Laboratory in Rochester, Minnesota:**

Eric N. Herness
Kevin Sutter

**IBM Development Laboratory in Santa Teresa, California:**

Harry Nayak
David Wisneski

**IBM Development Laboratory in Toronto, Canada:**

Chris Brealy
Tim Francis
Suman Kalia
Christina P. Lau
Yen Lu

**IBM International Technical Support Organization:**

Marcus Brewer, Editor
Eugene Deborin
Joe DeCarlo
Steve Gardner
Bob Haimowitz
Hanspeter Nagel

**IBM EMEA Application Development Software Centre, La Gaude:**

Jean Pierre Augias
Jean Michel Fauconnier
Joaquin Picon
Bruno Georges
Phiippe Gregoire

## Comments Welcome

**Your comments are important to us!**

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 83 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@us.ibm.com`

The "Ready" section takes you through all the steps to install the CBConnector Bank successfully. If the system is built up from scratch, you need to go through chapters 1 through 5. If a CBConnector environment has been installed, the chapters should still be skimmed through so that no particularities needed for the bank application setup are missed. Chapters 4 and 5 describe installation of the legacy tier on MVS and its replication on NT. If you install the legacy tier only on NT, you should still read chapter 4 because this is the part where the legacy application is described.

After the CBConnector environment has been installed, we recommend that you install the test applications described in the appendix, so that you can verify that your system has been set up correctly.

If you’re not a developer, but use this book to set up a CBConnector demo, the "Go" section provides a high-level understanding of the application architecture and the demo scenario. If, on the other hand, you are a developer, you have probably already been through these concepts in the *CBConnector Bank Implementation* redbook.

# Part 1. Ready

Part 1 guides you through the preparation of your CBConnector server and the legacy system. We also give you the chance to test your installation systems and subsystems before you install the CBConnector Bank application and provide a roadmap for the installation.

Please use the referenced product manuals as recommended is this publication.

# Chapter 1. Introduction

In today's financial services industry, customers demand fast and convenient access to their banking solutions. They want more variety in products and they want them to be delivered at low cost across a wide array of delivery channels. For example, banks must evolve and adapt to emerging customer needs, at low cost and without ignoring existing legacy applications.

The re-use of legacy systems is an issue that software architects cannot avoid. Software builders cannot afford to periodically rebuild their software from scratch. Many software designs are now produced by combining and elaborating existing architectural design fragments.

Work on architectures for software systems and studies for better ways to support software development have been around for quite a while. There is agreement now that Object Technology is the best basis for software architectures. The properties of object modeling, such as encapsulation and inheritance, facilitate the prototyping of new software and the integration of legacy applications into an object environment. The use of Object Technology considerably speeds up code maintenance and development time, consequently reducing the development cost. Therefore, Object Technology has become the de-facto standard for designing and building software.

One of the primary emphases of IBM's Component Broker Connector (CBConnector) is to support the evolution of the business enterprise. Its support for distribution and object modeling makes it the ideal tool for addressing evolving business market needs. The integration of legacy resource managers allows the leveraging of the huge resources invested in existing applications, including transactions and data.

We created a sample CBConnector Bank application that is delivered on the CD-ROM that accompanies this book. The CBConnector Bank application design and implementation is discussed in the *IBM CBConnector Cookbook Collection: CBConnector Bank Implementation,* SG24-5119*.*

This redbook provides you a roadmap for the preparation of your system and subsystems in order to install the CBConnector Bank server and client applications. This book does not provide a full description of CBConnector installation procedures. For more information, please read the *IBM CBConnector Quick Beginnings Guide.*

## 1.1 CBConnector Bank Server–Two Scenarios

The CBConnector Bank has two major scenarios:

1. **The Transient Customer Scenario** - the Customer Object is kept as a transient. The rest of the application objects are backed by UDB database residing on the middle-tier. This scenario is suitable for a portable installations, for instance on your ThinkPad.
2. **The CICS Customer Scenario** - the Customer Object is backed by CICS/DB2 either on mainframe or on Windows NT. The rest of the application objects are backed by UDB database residing on the middle-tier. This scenario is more robust and requires two tiers for its demonstration.

## 1.2 CBConnector Bank Client Two Scenarios

The CBConnector Bank client can be started in two versions:

1. **Bank teller** - offers a full range of available functions
2. **Home-Banking terminal** - offers a subset of functions, such as query of an account and transfers between two accounts

The full application scenario is described in Chapter 8.3.2, “A Complete Workflow Scenario” on page 50.

## 1.3 Organization of the CD-ROMs

The CD-ROMs have following main structure:

**CD-ROM # 1**

1. The *Book* subdirectory contains four CBConnector publications in PDF format:
   - *IBM CBConnector Overview*, SG24-2022
   - *IBM CBConnector Cookbook Collection: First Steps*, SG24-2033
   - *IBM CBConnector Cookbook Collection: CBConnector Bank User Guide*, SG24-5121
   - *IBM CBConnector Cookbook Collection: CBConnector Bank Implementation*, SG24-5119
2. CBBank subdirectory
   - **Client -** Client source code and interchange files

- **Install -** Installation directory
- **Middle Tier -** Object Builder models and source code for the CBConnector server applications for Transient Customer Scenario and Rational Rose models
- **TestCBC -** Two test Ping applications for C++ and Java

**CD-ROM # 2**

1. **CBBank**
   - **Install -** installation directory
   - **Legacy-** MVS and the replica for Windows NT source code
   - **Middle Tier -** Object Builder models and source code for the CBConnector server applications for CICS Customer Scenario

# Chapter 2. Hardware and Software Prerequisites

In this chapter, we describe the hardware and software prerequisite for the client workstation and for the CBConnector 1.2 server.

## 2.1 Windows NT Client

The processor, memory, and disk space requirements are the same as for the native operating system. The software prerequisites are:

1. Microsoft Windows NT 4.0 with Service Pack 3 applied
2. JDK 1.1.5 and above

## 2.2 IBM CBConnector Server on Windows NT Platform

This section describes the hardware prerequisites for the CBConnector server. To get complete information, please refer to *IBM CBConnector Quick Beginnings Guide.*

### 2.2.1 Hardware Requirements

This section lists the optimal system requirements for Component Broker.

Component Broker Server (runtime or development):

- 200 MHz processor or better
- At least 128 MB of memory
- At least 5 GB of disk space
- 800x600 capable display
- At least 300 MB of paging space software prerequisites

### 2.2.2 Software Requirements

The following products are prerequisites for Component Broker for Windows NT:

1. Microsoft Windows NT 4.0 with Service Pack 3 applied
2. Digital DCE 1.1c or IBM DCE 1.1.1 client (with FixPak DCENT11CE03 applied) with access to a DCE Security Server and a DCE Cell Directory Server
3. IBM DB2 Client CAE (Client Access Enablers for Windows NT and Windows 95, Version 4.0.1) Version 5.0 with access to an IBM Database Server Version 5.0

4. IBM VisualAge C++ for Windows, Version 3.5.3 with FixPak WTC354 applied
5. Javasoft Java Development Kit (JDK) 1.1.5
6. A frames-capable HTML 3.2 compatible browser to view the Component Broker online documentation.

**Notes**:

1. The DCE client and FixPak, if required, are included on the CBConnector Supplemental Programs compact disc. The DCE Security and Cell Directory Server are part of a separately licensed DCE package.
2. DCE is required for security-enabled clients only.
3. The Universal Database Client (UDB) client is included on the CBConnector Supplemental Programs compact disc. The Database Server is part of a separately licensed UDB package.
4. To develop Component Broker applications using UDB, the UDB SDK is required. The UDB SDK can be installed from the IBM Database Server compact disc.
5. The UDB SDK can be installed on any system with Component Broker, whether it is a UDB server or a client.
6. TCP/IP is a prerequisite for all Component Broker components. TCP/IP is installed as part of the operating system.
7. VisualAge C++ for Windows is a prerequisite for server application and CORBA C++ client development environments only and is included in the
8. Install VisualAge C++ for Windows and FixPak WTC354.

### 2.2.3 Installation Considerations

The installation considerations and the installation process is fully documented in the *Component Broker Connector - Quick Beginnings Guide*. This section includes consideration for planning your installation. Considerations include:

- **Installation**

Component Broker must be installed on a local drive. However, the install image can be accessed through a network drive. Before you install Component Broker, you need to plan your site to determine which Component Broker components to install on which computer. Because of software interactions, you must install the Component Broker software and prerequisites in the following order:

1. Microsoft Windows NT 4.0
2. UDB
3. VisualAge C++ for Windows
4. NT Service Pack 3
5. DCE
6. DCE FixPak DCENT11CE03
7. CBConnector package

- **DCE**

All Component Broker servers must have access to DCE 1.1c (also known as DCE 1.1.1) for object naming purposes. You can choose to install DCE on the same host as the System Manager, or you can install DCE on a remote host. You must determine which systems will be DCE Security and Name servers. You can install DCE on a single computer, or you can break it up into its component parts and install each server on separate computers. The DCE renting must be installed on each Component Broker computer. Dynamic IP addressing (DHCP) is not supported.

- **Universal Database (UDB)**

All Component Broker servers must have access to UDB. You can choose to install UDB on the same host as the System Manager, or you can install UDB on a remote host.

- **VisualAge C++ for Windows**

This must be installed on each computer intended for application development. After installing VisualAge C++ for Windows, you must install the Windows NT Service Pack 3 again.

- **Component Broker**

Please read the *Component Broker Connector - Quick Beginnings Guide* in order to designate a bootstrap host. A bootstrap host is a host from which remote client computers can bootstrap into the system name space. Every host computer that has the Component Broker server installed can be a bootstrap host. Decide which host computer each Component Broker computer will use for its bootstrap host. Not all client computers need to use the same bootstrap host. You should limit the number of computers that share the same bootstrap host to reduce the impact to your system if one of the bootstrap hosts should fail. Assign a port ID to use for each bootstrap host.

The port ID you supply during Component Broker installation must match the port ID configured with the corresponding bootstrap host in its Component Broker TCP/IP Protocol Image. The TCP/IP Protocol Image for a host can use the default System Management port ID (900). If you do not know what port ID to use, accept the default value, and change it later if that port ID is already in use. Designate one or more computers as Component Broker System Manager hosts. Each System Manager host manages one or more Component Broker clients and servers to form a Management Zone of systems. Each managed Component Broker client or server belongs to one Management Zone. During Component Broker installation, you need to provide the host name for the System Manager host system.

All host names defined during installation and management of Component Broker should be fully-qualified names. Determine locations for the System Manager User Interface. The System Manager User Interface can be installed on a computer other than the computer on which the System Manager is installed. Determine how you want applications to be developed and deployed. Applications can be deployed only from a server, but can be developed on either a server or a client. If you have a large site, you might not want a computer to share deployment and System Manager responsibilities, and you might want to mirror the applications to more than one server. Consider installing the System Manager as part of your first Component Broker installation. The System Manager is used to manage the other Component Broker servers and clients.

**Note:**

If installing Component Broker in a development environment, your first Component Broker installation must be a development server with VisualAge C++ from Windows installed as a prerequisite.

Consider performing the Server Package installation option on other computers that you want to be CBConnector servers. During the installation, you must provide the host name and port ID used for System Management services. Provide the information you specified during the typical install installation.

Consider installing the typical clients on hosts that you want to run client applications that use CBConnector applications. You must supply the System Manager host name and port ID and information about booster hosts. Provide the information you specified during the typical install installation. A frames-capable, HTML 3.2-compatible browser is required to view the CBConnector online documentation.

- **Web Server**

A Web server is required if Java Pallets that are developed to enable Java clients are going to be downloaded from the World Wide Web.

- **CBConnector Renting**

CBConnector Renting (server and client) comes with an imbedded Java Renting Environment (JRE 1.1.5). You only need the JDK if you want to develop client or server applications in Java, using either the Java client or using the CBConnector Object Model (SOM4) functions for developing C++ or Java applications. The Java 1.1.5 JDK for Windows NT can be obtained over the World Wide Web from the Javasoft home page.

Component Broker does not work with the Microsoft Java Renting environment.

- **InstallShield3**

The InstallShield3 development kit can be used to package applications generated by the Chocolaty package. InstallShield3 can be ordered separately from the InstallShield software corporation.

### 2.2.4 Installation Options and Required Prerequisites

Component Broker components installed from the CBConnector compact disc include:

**Typical Install**

This installation option allows you to define and manage servers as well as develop and deploy applications on your desktop. You should consider using this installation procedure for your first installation.

**Typical Clients**

This installation option installs the CORBA client, allowing the user to run existing Component Broker applications.

**Server Package**

This installation option includes the Component Broker server, the CORBA client, and the System Management Agent. This package is useful for a deployment system for applications that are to be developed and managed from other hosts.

**System Management Workstation**

This installation option includes the System Manager and the System Manager User Interface and defines a host that can be used to manage other CBConnector hosts.

**Custom Install all Available Packages**

This installation option enables you to select any or all of the Component Broker components for installation. Component Broker components installed from the Chocolaty compact disc include:

**Adding the Development Environment**

This installation option installs the Component Broker application development environment. You can specify to install development tools and software.

**Development Kits including:**

- CBCToolkit
- Server SDK
- CORBA Client SDK
- ActiveX Client SDK
- Java Client SDK
- CICS and IMS Application Adaptor SDK
- TKSamples

The following tables list the Component Broker components you can choose to install on your system from the CBConnector compact disc and the Collegiate compact disc and the prerequisites that apply to each component. The table below lists components and related prerequisites that you can install in a development environment.

*Table 1. Development Environment Prerequisites*

| Component | Windows NT4.0 | VisualAge C++ for Windows | Access to DCE | | Access to UDB |
|---|---|---|---|---|---|
| | | | Security | Directory | |
| Server | X | X | X | X | See Note 3 |
| CORBA Client | X | X | X | X | |
| Java Client | See Note 2 | X | X | X | |
| System Manager | X | | | | |

| Component | Windows NT4.0 | VisualAge C++ for Windows | Access to DCE | | Access to UDB |
|---|---|---|---|---|---|
| | | | Security | Directory | |
| System Manager User Interface | X | | | | |
| System Manager Agent | X | | | | |
| Application Adaptor | X | | | | |
| Documentation | | | | | |
| Samples | X | X | | | |
| Server SDK | X | X | | | |
| CORBA Client SDK | X | X | | | |
| Java Client JDK | See Note 2 | | | | |
| CBCToolkit | X | X | | | See Note 3 |

1. Visual Basic samples provided with ActiveX client require Microsoft Visual Basic, Version 4.0.
2. Any native operating system supported by Javasoft Java Virtual Machine 1.1.5.
3. The DB2 SDK is required for developing Component Broker applications.

The table below lists components and related prerequisites that you can install in a run-time environment.

*Table 2. Run-Time Environment Prerequisites*

| Component | Windows NT 4.0 | Access to DCE | | Access to DB2 |
|---|---|---|---|---|
| | | Security | Directory | |
| Server | X | X | X | X |
| CORBA Client | X | X | | |
| Java Client | See Note 1 | X | | |
| System Manager | X | | | |
| System Manager User Interface | X | | | |
| System Manager Agent | X | | | |
| CICS /IMS Application Adaptor | X | | | |

| Component | Windows NT 4.0 | Access to DCE | | Access to DB2 |
|---|---|---|---|---|
| | | Security | Directory | |
| Documentation | | | | |
| 1.Any native operating system supported by Javasoft Java Virtual Machine 1.1.5 | | | | |

## 2.3 Windows NT Replica of MVS Legacy System Server

We chose a separate physical tier for this installation.

### 2.3.1 Hardware Requirements

This section lists the optimal system requirements for Component Broker.

Component Broker Server (run time or development):

- 160MHz processor or better
- At least 128 MB of memory
- At least 2 GB of disk space
- 800x600 capable display
- At least 150 MB of paging space Software Prerequisites

### 2.3.2 Software Requirements

This section lists the software requirements for Component Broker.

Microsoft Windows NT 4.0 with Service Pack 3 applied

IBM DB2 Version 2.1

IBM TX Series for Windows NT Version 4.2

IBM DCE version 2 with ECO Pack 5 applied

IBM VisualAge COBOL Version 2.1

# Chapter 3. Installing the Client System and CBConnector Server

This chapter explains how to prepare your CBConnector server and client system before you install the CBConnector Bank application.

## 3.1 Installing the Base Windows NT System

The base Windows NT system installation is a standard NT installation using an Ethernet or token-ring network with TCP/IP connectivity. We recommend you follow these installation procedures or those in a publication written on this subject.

We also recommend you check the *system* and *user* variables after the installation. Specifically, in a Java environment, the CLASSPATH information has to be correct.

## 3.2 Preparing the Base Windows NT System

In order to prepare the client system for your CBConnector Bank installation, be sure that the JDK 1.1.5 is installed.

## 3.3 Installing CBConnector Server

In this section, we give you only the list of steps you have to go through in order to install the CBConnector server. Please precisely follow the *IBM Component Broker Connector - Quick Beginnings Guide* for a successful CBConnector server installation.

The installation steps are:

1. **DCE installation**:
   - Runtime
   - Cell Directory Server
   - Security Server
   - DCE Setup
2. **Universal Database Server Installation**:
   - UDB SDK
   - UDB Enterprise Edition
3. **CBConnector Server**
   - Typical install
   - Java client
   - CBConnector Toolkit (only in case you will be developing)

4. Start the CBConnector System Management.
5. Activate **Sample Configuration** in the **Sample Zone**.
6. Check if the Name Server is running properly.
7. Run createAllLoc.cmd from the CD-ROM's *CBBANK\TestCBC\Utility* subdirectory.
8. Install the two Ping test applications as described in Appendix A, "Test the Component Broker Connector Installation" on page 55. We recommend reading the full description of this installation in the *IBM CBConnector Cookbook Collection: First Steps*, SG24-2033.

# Chapter 4. MVS Legacy System Roadmap

This chapter provides information pertaining all the "legacy system" components required to do the necessary configuration and preparation of the MVS system. This includes the information required by the administrators of CICS and DB2.

The purpose of this chapter is not to explain how the various system administration functions should be done, but rather provide the information on what to use during the preparation phase. If any information is required on how the tasks have to be done, please refer to the appropriate system administration and installation guides.

## 4.1 Preparing the MVS Legacy System

Separate preparation phases are described for DB2, CICS and the COBOL application. We assume that the necessary CICS and DB2 base installations already exist prior to the preparation for the CBConnector Bank sample application.

### 4.1.1 DB2 Preparation

The DB2 definitions required for the sample application have been done using SPUFI. You, or your administrator, should use the method that complies with your installation standards or with the method you feel comfortable with. The source for the definitions referred to are included in the *\CBBANK\LEGACY\MVS\DB2* directory of the CD-ROM.

**Database Administrator:**

The sequence described below explains the DB2 preparation that was done during the development of the CBCBank sample application. We recommend that the same sequence be followed, unless the source code provided is modified to suit your installation standards.

The following DB2 definitions should be done in the sequence listed:

1. Storage group
2. Database
3. Tablespace
4. Tables and Indexes
5. Load test data

The various DB2 components and the corresponding files on the CD-ROM are listed in Table 3.

*Table 3. DB2 Components, Definitions and File Names*

| DB2 Component | DB2 Definition | File name ( *.txt ) |
|---|---|---|
| Storage Group | CBCBNKG1 | CBCSGDEF |
| Database | CBCBNKD1 | CBCDBDEF |
| Tablespaces | CBCBNKS1<br>CBCBNKS2<br>CBCBNKS3<br>CBCBNKS4<br>CBCBNKS5<br>CBCBNKS6<br>CBCBNKS7<br>CBCBNKS8 | CBCTSDEF |
| Tables and Indexes | CBCCUST<br>CBCADDR<br>CBCNOTE<br>CBCACNT<br>CBCTRAN<br>CBCPROD<br>CBCPPOL<br>CBCPLCY<br>CBCI1CUS<br>CBCI1ADD<br>CBCI1NOT<br>CBCI1ACN<br>CBCI1TRA<br>CBCI1PRO<br>CBCI1PPO<br>CBCI1PLC | CBCTBDEF |

Sample test data have been included on the CD-ROM in the same directory mentioned above. The test data is to be used during the testing phase as described in “Testing the Installation” on page 22.

The test data, file names containing the definitions, and the tables it relates to are described in Table 4.

*Table 4. Test Data Definitions*

| Test Data Definition | DB2 Table | Filename ( *.txt ) |
|---|---|---|
| Customer information | CBCCUST | CBCLDCUS |
| Address Information | CBCADDR | CBCLDADD |

These are the only two tables that test data are provided for and that will be used for testing the CBC installation.

**Note:** We recommend that the necessary privileges be granted to the users of the system as the final step of the DB2 preparation.

The COBOL declarations for the tables have been done using the DCLGEN facility with the parameters as described in Figure 1. These declarations need not be done by as part of the preparation phase, but are mentioned because the structures created by the declarations are used during the next preparation phase.

```
ACTION ..........===> REPLACE
COLUMN LABEL .....===> NO
STRUCTURE NAME ...===>
FIELD NAME PREFIX ===> 'XXXX-' (last 4 characters of tablename)
DELIMIT DBCS .....===> YES
COLUMN SUFFIX ....===> YES
INDICATOR VARS ...===> NO
```

*Figure 1. DCLGEN Parameters*

The structures generated by the declarations can also be found on the CD-ROM in the directory *\CBCBANK\LEGACY\MVS\COBOL*. The format of the file names are as follows: *CBCS*xxxx, where 'xxxx' = the last four characters of the table name the structure refers to.

### 4.1.2 Application Program Preparation

Preparing the application programs for installation consists of the following steps:

1. Copy the source code for the programs and copybooks, including the DB2 structures and BMS macro to the installation's preferred libraries.
2. Generate the BMS symbolic and physical maps and link-edit the BMS module.

3. Compile and Bind the DB2 programs

   "Compile" in this context refers to the whole process required to get the program from source code to object code in a state ready for execution, which includes translation, precompiling, actual compiling, and link-editing.

4. Compile CICS programs

The source code for the application programs and BMS maps are stored in the *\CBBANK\LEGACY\MVS\COBOL* directory. No compilation procedures are provided as part of this preparation phase. We assume that the person doing the preparation of the application programs will use the compile procedures provided by your installation. These compile procedures should make provision for including the correct source libraries, load libraries and system libraries used by the installation. It should also include the correct DB2 DBRM libraries, plan names, and bind procedures used by the organization.

Table 5 provides the names, short descriptions, and types of source code used in program preparation.

*Table 5. Application Program Preparation*

| Name | Description | Type |
|---|---|---|
| CBCSET1 | BMS Mapset Source | CICS BMS |
| CBCSET | BMS Symbolic Map (generated) | COPYBOOK |
| CBCPGC01 | CBCBank Menu Program | CICS COBOL II |
| CBCPGCUS | Customer and Address Information Program | CICS COBOL II |
| CBCIOCUS | Customer Database Input/Output Program | COBOL II DB2 |
| CBCIOADD | Address Database Input/Output Program | COBOL II DB2 |
| CBCCOMMA | CICS Communication area - General Info. | COPYBOOK |
| CBCCACUS | CICS Communication area - Customer Info. | COPYBOOK |
| CBCCAADD | CICS Communication area - Address Info. | COPYBOOK |
| CBCSCWA | Common Work Area for COBOL programs | COPYBOOK |
| CBCSxxxx<br><br>CBCSCUST(1)<br>CBCSADDR | DB2 Table Structures<br>xxxx = last 4 characters of DB2 table name<br><br>Customer Information<br>Address Information | COPYBOOKS |

| Name | Description | Type |
|---|---|---|
| 1.These are the only two DB2 structures required by the application programs. | | |

The intention is not to implement a fully functioning banking system, but rather to provide functionality on the MVS system that can demonstrate the use of Component Broker with the Procedural Application Adaptor (PAA) and the CICON tool. Therefore, these are the only application programs implemented for the legacy system on MVS.

## 4.2 Installing CBConnector Bank Transaction

The installation of the systems on MVS involves doing the CICS system definitions. As with the DB2 definitions in the previous chapter, we provide the information required by the CICS system administrator for defining the sample application to CICS. We assume that the person doing the installation has a working knowledge of CICS system administration.

Refer to Table 6 for the various CICS system components that have to be defined, which include:

1. Defining a Group for the new system, if not adding the definitions to an existing group. The rest of the components will be defined as part of this group.
2. Mapset for the CICS screens
3. Application programs
4. Transaction

*Table 6. CICS System Definitions*

| CICS Component | Name |
|---|---|
| Group | CBCGROUP |
| Mapset | CBCSET |
| Programs | CBCPGC01<br>CBCPGCUS<br>CBCIOCUS<br>CBCIOADD |
| Transaction | CBC1<br>(Link to program CBCPGC01) |

## 4.3 Testing the Installation

On completion of the previous two preparation phases, the installation should be ready to be tested. The test data provided includes 11 customer records with address information.

The installation can be tested by using transaction code `CBC1`.

The menu screen for the CBCBank application should be presented as shown in Figure 2. Select the **Customer Detail** function by keying in `CUST` in the space provided.

```
CBCBANK : MENU

    CUSTOMER DETAIL      - CUST

    CUSTOMER NOTES       - NOTE

    ACCOUNT DETAIL       - ACCT

    TRANSACTION DETAIL   - TRAN

    PRODUCT DETAIL       - PROD

    PRODUCT POLICY       - PPOL

    POLICY DETAIL        - PLCY









 1 - SELECT A FUNCTION FROM THE LIST SUPPLIED
```

*Figure 2. CBCBank Menu Screen*

The next screen to be shown is the Customer Detail Screen, as shown in Figure 3.

```
CBCBANK
                        CUSTOMER INFORMATION
 ACTION:   (1-CREATE, 2-READ, 3-UPDATE, 4-DELETE)
 KEY         :
 TITLE       :
 LASTNAME    :
 FIRST NAME  :
 ID NUMBER   :
 DATE OF BIRTH:                          GENDER:    (M / F)
 HOME PHONE  :                           WORK PHONE:
 HOME ADDRESS :




 POST ADDRESS :



 1 - SELECT A FUNCTION FROM THE LIST SUPPLIED
```

*Figure 3. Customer Detail Input Screen*

On this screen, select the **READ** option by keying in a `2` in the `ACTION` field and any key in the range from `7000001` to `7000011` on the `KEY` field. This invokes the following programs:

1. CBCPGCUS
2. CBCIOCUS
3. CBCIOADD

These programs will select the information from the CBCCUST and CBCADDR DB2 tables and display an output screen containing the customer information required.

```
CBCBANK
                        CUSTOMER INFORMATION
 ACTION:   (1-CREATE, 2-READ, 3-UPDATE, 4-DELETE)
 KEY         :                        7000004
 TITLE       : MR.
 LASTNAME    : MARK
 FIRST NAME  : FITZPATRICK
 ID NUMBER   : 40000000000000000004
 DATE OF BIRTH: 1956-04-23              GENDER: M  (M / F)
 HOME PHONE   : (004) 789-555-567890        WORK PHONE: (004)
987-555-567890
 HOME ADDRESS :
 7000004 HAD STREET
 EZE
 BRISBANE
 4444444444
 AUSTRALIA

 POST ADDRESS :
 PO BOX 8908976
 SOMEWHERE IN AUSTRALIA
 797897893
 13312

 0 - FUNCTION SUCCESSFUL: SELECT NEW ACTION OR CLEAR
```

*Figure 4. Customer Detail: Information Screen*

If a screen is returned which resembles the one shown in Figure 4, it signifies that the installation was successful.

## 4.4 Summary

The preceding chapter served as an explanation of the information required by administrators of DB2 and CICS systems when preparing the MVS legacy system. The intention was not to give a full explanation of the functionality of the systems or that of DB2 or CICS, but rather to serve as a roadmap that they could use.

It is important that you not only follow the standards of your installation but also adhere to the naming standards and conventions of the sample application.

This preparation enables the MVS system for use by the other tiers of the CBCBank sample application.

# Chapter 5. Replicating the Legacy System to Windows NT

This chapter describes how to replicate the legacy system onto the NT platform.

The purpose of this chapter is therefore not to give you a guided tour of the inner workings of CICS, DB2 or DCE, but rather to serve as a short roadmap leading to the information about the legacy software. You do not have to physically download any software from the MVS system; everything you require for replication is provided on the CD-ROM.

Following this roadmap should assist you in setting up the legacy application on NT.

## 5.1 Preparing for Replication of the Legacy System

The replication of the legacy system is by no means connected with the CBConnector Server. It is a stand-alone server, in other words, a separate physical tier used as a legacy system server. Figure 5 indicates the software that should be installed on your legacy system server before attempting the replication:

Please, notice the specific release numbers.

| |
|---|
| Microsoft Windows NT 4.0 with Service Pack 3 applied |
| IBM DB2 Version 2.1 |
| IBM TX Series for Windows NT Version 4.2 |
| IBM DCE version 2 with ECO Pack 5 applied |
| IBM VisualAge COBOL Version 2.1 |

*Figure 5. Software Requirements for a Replica of the Legacy Server*

**TXSeries Quick Beginnings:**

If you are not familiar with installing and configuring TXSeries, it is definitely worth the effort of referring to the *Quick Beginnings* guide supplied as part of the online documentation. It is especially useful for configuring DCE and CICS and for setting up DB2 as the CICS file manager.

### 5.1.1 Sequence of Software Installation

We installed the software in the following order:

1. Windows NT
2. TXSeries (CICS)
3. DB2
4. Configuration of DCE for use with CICS
5. Configuration of CICS

### 5.1.2 DB2 Preparation

We assume that you already have the correct version of DB2 installed and that you have a working knowledge of DB2.

- If you selected, as we have, using DB2 as opposed to SFS for the CICS resource definition files, ensure that you have DB2 installed and your local database defined before attempting to configure CICS.
- Ensure that you have included the following setting in your environment variables:

  ```
  DB2COMM = TCPIP
  ```

- We once again suggest that you follow the steps provided in the TXSeries *Quick Beginnings* guide for the configuration of DB2 for use with CICS. This provides you with step-by-step instructions.
- Make sure that you connect to your local database after completing the configuration steps.

### 5.1.3 DCE Preparation

You should consider following when configuring DCE for CICS:

- Make sure that you do not use a different user identification than the default DCE administrator `cell_admin` when running `cicssetupdce`. The program has a validation routine that insists you be logged on as the above-mentioned user ID. If you have already installed DCE with a different administrator ID, the next best thing is cloning your administrator ID, but ensure that the `object creation quota` for `cell_admin` is changed from its default `cloning size` of `0` to `> 0`.
- If you have problems with user messages not being displayed properly, ensure that you have included the *EN_US* directory on the `NLSPATH` of the environment variables.

### 5.1.4 CICS Preparation

You probably do not have to do anything in addition to what you have already done when you installed CICS. For completeness, we mention those things that we did to configure CICS. If you have not done any of the steps below, we suggest that you spend some time doing them now.

The following configuration steps have to be done prior to installing the legacy system on NT:

- Setting up a file manager for CICS using a DB2 database. Refer to “DB2 Preparation” on page 28.
- Configuring a CICS region. You probably already have one or more CICS regions defined. We suggest that you ensure that the correct settings as given in TXSeries *Quick Beginnings - Configuring a CICS Region* for the file system, DB2 server, DB2 instance, and default user ID and password that have been provided.
- You will also find it handy to install a CICS Telnet server for the region if you have not done so already.
- Configuring a listener process for the CICS region. We have configured a listener process using TCP/IP, enabling a CICS Client on another separate physical device to access the region defined on the CICS server. We selected the default CICS user identification `CICSUSER`.
- Prepare your CICS client to be ready to access your CICS region. You can install the CICS client on your CICS server or on another system that has TCP/IP connectivity. The following has to be change in your *CICSCLI.INI* file:

```
Server = CICS01
Protocol= TCPIP
NetName= IP address of your CICS server
Port= 0
```

### 5.1.5 Installing the Replicated Legacy System

After you have completed all of the phases required to prepare for replicating the legacy system, you should be ready to install the legacy system on the NT Server.

We have already copied all the required components from the MVS system. These files are available on the CD-ROM in *\CBBANK\LEGACY\NT\INSTALL*. An explanation of the files and what components are contained in them is shown in Table 7 on page 30.

The installation phase consists of a three distinct steps:

1. Installing the DB2 tables
2. Preparing the COBOL, CICS and DB2 programs
3. Defining the legacy application to CICS

We once again assume a working knowledge of application development using CICS, COBOL and DB2. We therefore only provide the information required for installing the application and do not explain exactly how you should do it.

*Table 7. Description of Installation Files*

| Filename | Description of Contents |
|---|---|
| CBCDB2.ddl | Table definitions and test data for DB2 |
| CBCMAKE.txt | Makefile for all COBOL, CICS and DB2 programs |
| CBCCICS.txt | CICS transaction: program and map definitions |

### 5.1.6 Installing DB2 for the Legacy System on NT

You should at this stage already have a DB2 system installed. We have provided the source code for the creation of the DB2 tables and the test data in the *CBCDB2.ddl* file in the *\CBBANK\LEGACY\NT\INSTALL* directory of the CD-ROM. We have specifically not included a database definition because you might already have a database defined that you may prefer to use. You must ensure that you use this database name when preparing the COBOL DB2 programs.

If you do not have a database defined, create a database called CBCBNKD1. The reason for the name is that it will make it easier for you during the next stage of preparing the COBOL programs. You do not have to change the database name referred to in the makefile.

You can execute the *CBCDB2.dll* file, which creates the Customer and Address DB2 tables, using your DB2 command prompt. This file also populates the tables with test data.

### 5.1.7 Installing the Legacy System Programs on NT

The programs for the legacy system that have been downloaded are contained in the *\CBBANK\LEGACY\NT\COBOL* directory on the CD-ROM. The copybooks used by these programs are also in the same directory. For a

list of the actual source files and program descriptions and types, refer to Table 8.

*Table 8. Legacy System Source Files*

| File name | Description |
|---|---|
| CBCSET1.bms | BMS Mapset Source |
| CBCPGC01.cob | CBCBank Menu Program |
| CBCPGCUS.cob | Customer and Address Information Program |
| CBCIOCUS.cob | Customer Database Input/Output Program |
| CBCIOADD.cob | Address Database Input/Output Program |
| CBCCOMMA.cpy | CICS Communication area - General Info |
| CBCCACUS.cpy | CICS Communication area - Customer Info |
| CBCCAADD.cpy | CICS Communication area - Address Info |
| CBCSCWA.cpy | Common Work Area for COBOL programs |
| CBCSCUST.cpy | Customer Information DB2 Copybook |
| CBCSADDR.cpy | Address Information DB2 Copybook |

These files are used in the makefile, *CBCMAKE.txt*. You must ensure that the makefile is modified to refer to the correct DB2 database.

Copy the CBCMAKE.txt, renaming it CNCMAKE.MK, to your COBOL development directory. Copy the CICS copy files from OPT\CICS\INCLUDE to your development environment. These are the files:

```
DFHBMSCA
DFHAID
CICS-API
CICS-EIH
```

Run the NMAKE utility from the DB2 command prompt by typing the following on the command line:

```
nmake -f cbcmake.mk
```

The following actions are performed:

- Generation of BMS physical and logical maps
- Precompile, compile, link-edit and bind of COBOL DB2 programs
- Translate, compile and link-edit of CICS COBOL programs

### 5.1.8 Defining the Legacy Application to CICS on NT

After the completion of the previous two steps, you still have to define the legacy applications to CICS. We elected to use the batch file method by entering the `cicsadd` command.

The *CBCCICS.txt* file contains the code for these definitions. You have to modify the source file to point to your own region that you have defined in CICS and ensure that the correct path is supplied before executing the file. If you prefer, you can also do the definitions using the administration utility supplied with CICS.

The following definitions are done:

- Map CBCSET
- Programs CBCPGC01, CBCPGCUS, CBCIOCUS, and CBCIOADD
- Transaction CBC1

### 5.1.9 Starting the Telnet Server Daemon

In order to use the HOD connection from the CBConnector server or from the VA Java with CICON environment, issue the following command from your command line:

```
CICSCP CREATE TELNET _SERVER cbcon -P 7777 -r scscpaa5
```

where `cbcon` is the Telnet server name, `7777` is the port number and `scscpaa5` is name of your CICS region.

### 5.1.10 Testing the Installation

If you have succeeded in getting this far, you are probably eager to see if your installation of the legacy system on NT actually works. From your local CICS terminal, enter `CBC1`, the name of our transaction.

Upon successful completion of this chapter, you should be ready for using the legacy system component of the CBConnector Bank application.

# Chapter 6. Installing CBConnector Bank Server Applications

This chapter provides instruction on how to prepare your CBConnector server system before you can install the CBConnector Bank application.

## 6.1 Installing CBConnector Bank

This section describes the installation of the CBConnector Bank server applications.

### 6.1.1 Creating and Binding the Databases

We provide you with two files in order to create the necessary databases. You can find them in the *CBBANK\INSTALL\DB2* subdirectory on your CD-ROM, together with the bind files. The *<drive:>**<directory>* means the *CBBANK\INSTALL\DB2* directory.

1. Database Name: **POLICY**
   - To create the Policy database, run the following command from the DB2 command line prompt:

     ```
     <drive>:\<directory>db2 -f createPolicy.ddl
     ```

   - From the DB2 command line prompt, execute the `bind` command:

     ```
     <drive>:\<directory>db2 bind PolicyPO.bnd
     ```

   - From the *<drive:>**<cbroker>**etc* directory, bind the following files:

     ```
     <drive>:\<cbroker\etc>db2 bind db2cntcs.bnd
     <drive>:\<cbroker\etc>db2 bind db2cntr.bnd
     ```

2. Database Name: **CBBANK**
   - To create the Policy database, run the following command from the DB2 command line prompt:

     ```
     <drive>:\<directory>db2 -tf createCbbank.ddl
     ```

   - From the *<drive:>**<directory>* directory, bind the following files:

     ```
     <drive>:\<directory>db2 bind notepo.bnd
     <drive>:\<directory>db2 bind tellerpo.bnd
     <drive>:\<directory>db2 bind checkaccountpo.bnd
     <drive>:\<directory>db2 bind savingsaccountpo.bnd
     <drive>:\<directory>db2 bind txnrecordpo.bnd
     ```

   - From the *<drive:>**<cbroker>**etc* directory, bind the following files:

     ```
     <drive>:\<cbroker\etc>db2 bind db2cntcs.bnd
     <drive>:\<cbroker\etc>db2 bind db2cntr.bnd
     ```

### 6.1.2 Transient Customer Scenario

This section contains information on how to install the CBConnector Bank Transient Customer Scenario. Follow these steps to successfully install the server application:

#### 6.1.2.1 Installing the Applications from CD-ROM # 1:

1. Install the CBBankPolicyApp by executing *setup.exe* from

   *CBBank**Install**CBBankPolicyAppFam**setup.exe*

2. Install the CBBankApp by executing *setup.exe* from

   *CBBank**Install**Transient Customer**CBBankAppFam**setup.exe*

3. Install the CBBankCustomerApp by executing *setup.exe* from

   *CBBank**Install**Transient Customer**CBBankCustomerAppFam**setup.exe*

#### 6.1.2.2 Using CBConnector System Management

In this section, we briefly guide you through the CBConnector System Management User Interface. These steps have to be taken in order to configure, activate and run the CBConnector Bank server applications:

1. Create a new **Management Zone** (arbitrary name).
2. Create a new **Configuration** in the Management Zone (arbitrary name).
3. From **Host Images->Application Family Installs**, locate the three folders, CBBankPolicyAppFam, Specific CBBankPolicyAppFam and iDB2IMApplications, and drag the following applications into your configuration:
   - CBBankPolicyApp
   - Specific CBBankPolicyApp
   - iDB2IMServices
4. From your Configuration, create a new **Free-standing server**, **CBBankPolicyServer**.
5. Drag the configured applications from your configuration to your new server.
6. From **Host Images->Application Family Installs**, locate the four folders, CBBankAppFam, Specific CBBankAppFam, CBBankCustomerAppFam and iDB2IMApplications, and drag the following applications into your configuration:
   - CBBankApp
   - Specific CBBankApp

- CBBankCustomerApp
- iDB2IMServices

7. From your configuration, create a new **Free-standing server**, **CBBankServer**.
8. Drag the configured applications from your Configuration to your new server.
9. Drag your two configured servers onto **Host -> Your machine**.
10. **Activate** your configuration zone.
11. From **Hosts Images -> Server Images Stop Immediate** your two servers.
12. Expand the servers and from **XA-Manager** and edit the databases with your user ID and password.
13. **Run Immediate** your servers.

After these steps, you are ready to install and run your CBConnector Bank client application.

### 6.1.3 CICS Customer Scenario

This section provides information on how to install the CBConnector Bank CICS Customer Scenario. Follow these steps to successfully install the server application:

#### 6.1.3.1 Installing the Applications from CD-ROM # 2:

1. Install CBBankPolicyApp by executing *setup.exe* from

   *CBBank**Install**CBBankPolicyAppFam**setup.exe*
2. Install CBBankApp by executing *setup.exe* from

   *CBBank**Install**CICS Customer**CBBankAppFam**setup.exe*
3. Install CBBankCustomerApp by executing *setup.exe* from

   *CBBank**Install**CICS Customer**CBBankCustomerAppFam**setup.exe*

#### 6.1.3.2 Using CBConnector System Management

In this section, we briefly guide you through the CBConnector System Management User Interface. These steps have to be taken in order to configure, activate and run the CBConnector Bank server applications:

1. Create a new **Management Zone** (arbitrary name).
2. Create a new **Configuration** in the Management Zone (arbitrary name).

3. From **Host Images->Application Family Installs**, locate the three folders, CBBankPolicyAppFam, Specific CBBankPolicyAppFam and iDB2IMApplications, and drag the following applications into your configuration:
   - CBBankPolicyApp
   - Specific CBBankPolicyApp
   - iDB2IMServices
4. From your configuration, create a new **Free-standing server**, **CBBankPolicyServer**.
5. Drag the configured applications from your configuration to your new server.

From **Host Images->Application Family Installs**, locate the five folders, ->CBBankAppFam, Specific CBBankAppFam, CBBankCustomerAppFam, iDB2IMApplications and iPAAApplications, and drag the following applications into your configuration

- CBBankApp
- Specific CBBankApp
- CBBankCustomerApp
- iDB2IMServices
- iPAAServices

6. From your configuration, create a new **Free-standing server**, **CBBankServer**.
7. Drag the configured applications from your configuration to your new server.
8. Drag your two configured server onto **Host -> Your machine**.
9. Activate your configuaration zone.
10. From **Hosts Images -> Server Images Stop Immediate** your two servers.
11. Expand the servers and from **XA-Manager** and edit the databases with your user ID and password.
12. **Run Immediate** your servers.

#### 6.1.3.3 Configuring Environment Variables (System and User)

The following environment variables have to be set:

1. CLASSPATH

   *<drive>:\<CBBankCustomerAppFam >\bin\bmsbean.jar*

<d*rive>:*\<*CBBankCustomerAppFam* >*bin*

2. PATH

<dr*ive>:*\<*CBBankCustomerAppFam* >*bin**bmsbean.jar*

*<drive>:*\<*CBBankCustomerAppFam* >*bin*

**Note**:

The names of the servers are arbitrary. CBBankPolicyServer can be named anything you like, but if you change it, you must update the row in the POLICY table in the POLICY database for the `dev/CBBankPolicy/PolicyManager/queryServer` entry.

# Chapter 7. Installing the CBConnector Bank Client Application

This chapter gives you the necessary information for installing and running the CBConnector Bank client application.

## 7.1 Installing CBConnector Bank Client

The recommended way for CBConnector Bank installation is to execute the *SETUP.EXE* from the *CBBANK\INSTALL* subdirectory on CD-ROM # 1. When the dialog for the application install appears on the screen, make a selection of the CBConnector Bank client application.

The next step will be to ensure that your CLASSPATH is updated with correct placement of the *CBCBank.jar* file and the somojor.zip. You can find these two files in the *<drive>:\<directory>* you specified during the installation process. In the same directory, you will also find two command files for your startup procedure:

*runCBCBank.cmd* (Bank teller application)

*applet.cmd* (Home-banking scenario).

## 7.2 Configuring CBConnector Bank Client

In order to configure the client application and make a choice of what scenario (Transient customer/CICS-backed customer) is to be executed, we are parsing an *.ini* file at application startup. Make sure this file is updated with following keywords:

```
CBVersion=1.2 (default) or 1.3
CustomerMode=CICS or Transient (default)
```

You can find the file in the directory specified during the installation, as described in 7.1, “Installing CBConnector Bank Client” on page 39.

## 7.3 CBConnector Bank as a Bank Teller Application

The following steps should be carried out to start the CBConnector Bank application as a stand-alone Java application:

1. Make the *<drive>:\<directory>* the current directory.
2. Ensure that your CBConnector Bank server applications are started.
3. Start the *runCBCBank* command file.

4. At the logon window, type in `teller` for the teller and `demo` for the password (take care to use lowercase letters only).
5. When the teller GUI appears, you may start using the functions.

## 7.4 CBConnector Bank as a Home-Banking Application

The following steps should be carried out to run the CBConnector Bank application as a Java applet within an applet viewer:

1. Make the *<drive>:**<directory>* the current directory.
2. Ensure that your CBConnector Bank applications are running.
3. Start the *applet* batch file.
4. There is no logon procedure for the home banker. However, only a subset of the functions available to the teller are presented and available to a home banker.
5. When the home banking GUI appears, you may start using the functions.

# Part 2. Go

After you have successfully installed the necessary system and subsystems in the previous chapters, and you have tested your installations, you are ready to run the CBConnector Bank application. Part 3 guides you through the scenario and explains what is demonstrated in the application.

Good luck!

# Chapter 8. Application Scenario

We provide the CBConnector Bank, which is a sample bank application, to demonstrate the CBConnector capabilities.

The objective for this scenario is to show how an end-user at a client on a workstation can access objects on a middle tier server. These objects retrieve the appropriate data from a back-end CICS system.

The application is *not* meant as a complete banking application with all the functionality needed for a bank's everyday operation. It's not even meant to look good. Its purpose is to show that an end user on a workstation can access different data stores on different platforms by using Component Broker.

This chapter takes you through the basic concepts of our simple banking application. We start by presenting the client application and the functionality it provides. Then, we introduce the different parts that make up the application architecture. After that, from a technical point of view, we follow one operation on its journey from the client, through the server to the back-end, and back. Finally, we give step-by-step instructions on how to operate the client for a complete demo.

The sample bank application implementation details may be found in the *CBConnector Bank Implementation,* SG24-5119, redbook. A technical overview of Component Broker is provided in the *IBM Component Broker Overview,* SG24-2022, redbook.

## 8.1 The CBConnector Bank

The CBConnector Bank application has functionality taken from the banking domain. It is only a small subset of an all-encompassing, real-world banking application. Even so, we hope it will give you an idea of how such an application looks when implemented in Component Broker, and it definitely demonstrates that a client can access a back-end legacy system by going through a Component Broker middle tier.

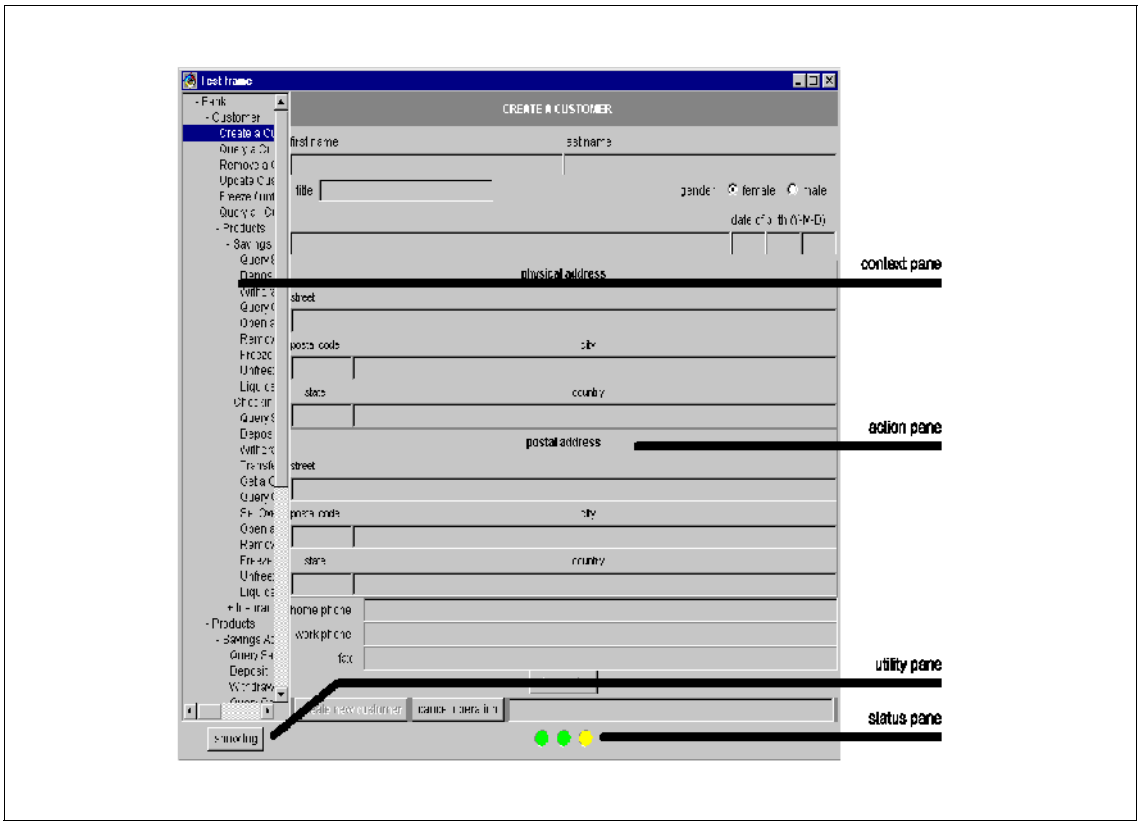



*Figure 6. The CBConnector Bank User Interface*

### 8.1.1 The User Interface

The layout of the interface is divided into two main parts. There is a set of actions presented as a vertical menu on the left that lists all the actions you may perform in this application. When you select an action to perform, the action requires user input. It presents a pane to the right of the menu, with fields the user can fill in. At the bottom of the pane are buttons that will trigger the action when pressed. The buttons are only enabled when all the required information has been filled in.

When an action is triggered, the pane will “freeze” until the action has been performed successfully. A status bar at the very bottom will indicate how the operation is doing. Yellow indicates that the operation is in progress, green that it has performed successfully, and red that is has failed.

Even though you have triggered an action that is pending, you are free to execute any other operation from the menu on the left (but not the same one). A second status indicator will pop up in the status bar on the bottom. By pressing the different status icons, you may switch between opened panes.

On the pane, there is a status field that, in text form, gives you information on how your operation is progressing. If this is not sufficient, you may look at the log pane, which gives you detailed descriptions of the transaction's movements during the operation. The button to show the log is at the very bottom of the application window.

### 8.1.2 The Actions

You may create and update customers, accounts, and transfer money between accounts. Below, you will find a list of the actual operations that may be performed, with a short description. These actions correspond to the ones you find in the left-side menu in the application user interface.

#### 8.1.2.1 Operations on Customer

**Create a customer.** You may input customer information, such as name and address, and create a new customer with this information.

**Query a customer**. You may find information about a certain customer by entering the customer's name.

**Remove a customer.** You may remove a customer that has been created by entering the name of the customer you want to remove.

**Update customer information.** First, you must find a customer, and then you can update information on it.

**Freeze or unfreeze all of a customer's accounts**. You may freeze all a customer's accounts in one operation, or, if they previously have been frozen, unfreeze them.

**Query all of a customer's accounts.** You may see all the accounts a customer has in this bank.

#### 8.1.2.2 Operations on Checking Account

**Query a checking account**. You may list all the transactions performed on a certain account by providing the checking account key.

**Deposit to a checking account**. You may deposit money to a checking account. First, find the account by supplying the key; then input the amount of money to deposit.

**Withdraw from a checking account.** Likewise, you may withdraw money from an account.

**Transfer from a checking account**. To transfer money between accounts, supply an additional account key to represent the account you want to transfer money to.

**Open a checking account.** Open an account by supplying account information. A key is provided by the system and must be kept to query this account later on.

**Remove a checking account.** By supplying the account key, you may remove an account.

**Freeze a checking account.** By supplying the account key, you may freeze an account.

**Unfreeze a checking account.** By supplying the account key, you may unfreeze an account.

#### 8.1.2.3 Operations on Savings Account

The savings account menu is identical to the checking account menu; the only difference is the type of accounts you handle.

## 8.2 Application Architecture

The application is divided into three tiers. The client tier is typically a workstation, where the client application resides. The client tier only handles user requests and passes them on to the middle tier. It is the middle tier that handles all the business logic. The business logic is implemented with Component Broker Objects on the middle tier server, which in our case is an NT server.

When the Component Broker objects need to retrieve and update data, they do so by interacting with a DB2 database on the NT server, as well as the legacy application on the legacy tier. The legacy tier is either an NT server or a mainframe with MVS, and it contains the CICS legacy application.

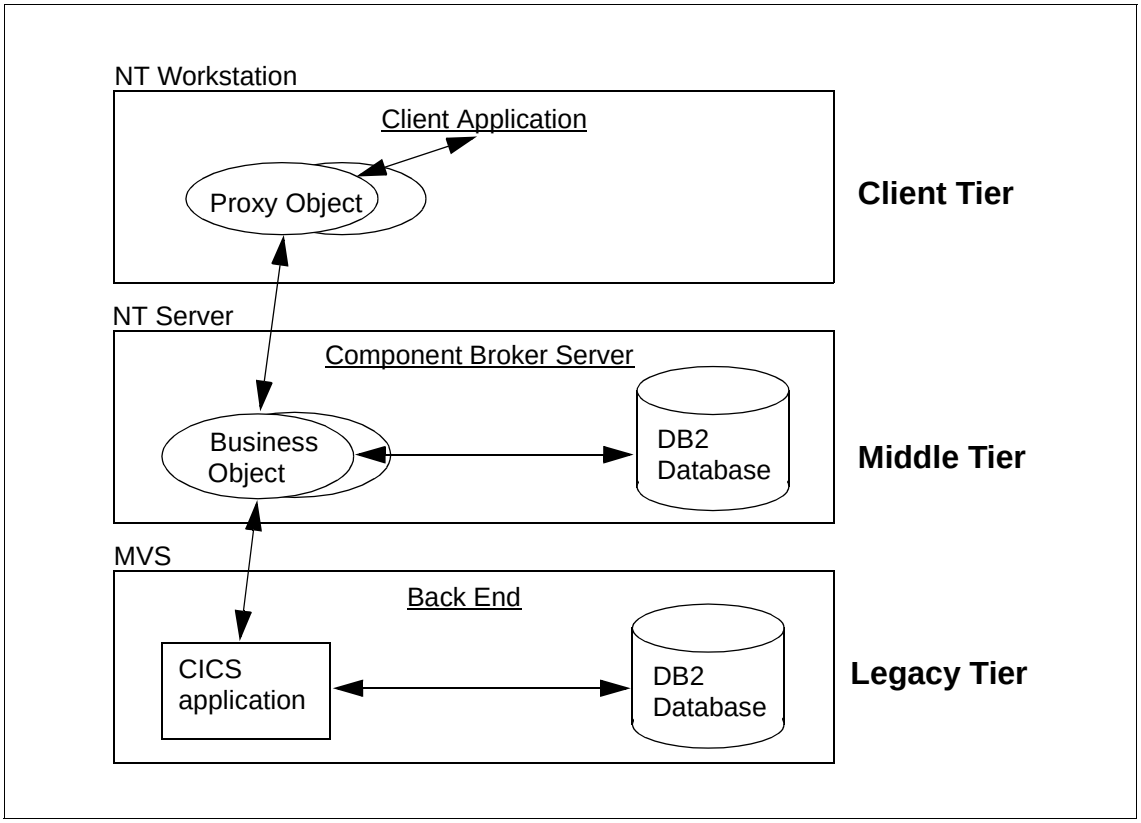



*Figure 7. The CBCBank Sample Application Framework*

### 8.2.1 Client Tier

The client tier is the interface to the end user. The user performs actions, and the system provides feedback. However, this is only the tip of the iceberg. When the user wants to perform an action, such as updating customer information, the client application takes the request and passes it on to the Business Object on the middle tier, where all the business logic processing occurs.

The client application doesn't send the information directly to the Business Object, though, because it would have to know about a lot of Component Broker-specific things it would rather disregard. Instead, it talks to a Proxy Object that acts as a broker between the client application and the Business Objects on the Component Broker Server. The Proxy Object takes the client request, establishes contact with the correct Business Object home on the middle tier, and passes the request forward in a language Component Broker

understands. It waits for a reply and passes information back to the client in a format the client can read.

### 8.2.2 The Middle Tier

The middle tier is the Component Broker server. The server is the home of the Business Objects that contain business logic and a persistence framework to store the information the objects hold.

A Business Object corresponds to an entity in the banking domain. There are Customer objects, CheckingAccount objects, and Transaction Objects, to name a few. The objects store information about themselves in different data stores. The Customer-related information is stored on the back-end and is retrieved and updated by calling CICS transactions. The other objects are stored in a DB2 database on the middle tier.

### 8.2.3 The Legacy Tier

The legacy tier contains the CICS application. This is usually a legacy system with information we want to re-use, and it acts as a transactional data store. The legacy tier can be CICS on a mainframe running MVS or CICS installed on an NT server.

## 8.3 The CBCBank Sample Application Scenario

Now that we have peeked into the architecture of the CBConnector Bank, we are ready to put the application to work.

First, we present one single action and reveal to you what actually happens behind the scenes when this action is performed. After that, we propose a complete demo of the application by combining actions to make up a sample workflow.

### 8.3.1 The Create Customer Scenario

We now demonstrate one full action in detail. We have been very unimaginative and have selected the first action on the list, namely Create Customer.

To start the CBCBank sample application, execute the following command in the directory where you installed the CBConnector Bank:

```
java CBCBank CommonDataContainer.ini 5
```

Or from a browser:

**open -> file ->** *<drive>:<directory> CBCBank.html*

Before any actions can be performed, the client needs to establish contact with Component Broker on the middle tier. It does so by providing a host name and port number to the father of all the proxies, the base proxy, and tells it to initialize everything. The base proxy then talks to an Object Request Broker (ORB), which is the glue between the proxy on the client side and Component Broker on the server, and initializes the connection.

A small dialog pops up prompting for teller name and password. The user is `teller` and the password is `demo`.

When the client application is up and running, the initialization has been done; the logon was successful, and you're ready to work.

You want to create a customer; so select the **create customer** action from the Customer menu on the left. A pane will pop up to the right, with customer information entry fields. You need to fill in the first and last name, select gender, date of birth, and the physical address. When the pane has been filled out sufficiently, the Create New Customer button will be enabled. Press it to create the customer.

Now, you wait until the yellow status bar turns green, and hope that it doesn't turn red. Maybe you even want to look at the log to see strange messages fly by while you wait. To enlighten you a little, here is a high-level description of what goes on.

When you request to create a customer, the client creates a proxy for the Customer Business Object on the server. The proxy sets up a connection to the home of Customer Objects on the Component Broker server. This home can create new Customer Objects as well as retrieve, update and delete objects. The home is also responsible for storing the objects in a persistent store.

The client fills the customer proxy with the information in the user interface. Then it tells the proxy to create itself, turns yellow, and patiently waits for an answer. The proxy takes the customer information and passes it to the Customer Business Object home and requests that such a Business Object is created in the home.

If an object is created in the Customer home, its counterpart needs to be created in a persistent store. We need a persistent store for the objects, so that when the Component Broker server stops, the objects live on. The persistence framework has **create**, **retrieve**, **update** and **delete** methods defined; so the home passes the **create** method to the persistence

framework, and it handles the create operation. The customer is stored in a CICS application; so this is where the framework creates the object. The persistent store could have been a relational database, and on a different server, with no change in the proxy or the Business Object.

When the customer has been successfully created in the CICS application, the Component Broker server tells the proxy that everything went OK. The proxy informs the client, in turn, and the status bar turns green. If any of the steps failed, the client is informed of the bad news, and the status bar turns red, accompanied by a message.

For all operations, the procedure is more or less the same. When you want to find a customer, the client asks for a new customer proxy that establishes contact with the customer home. The client puts the name of the customer in the proxy and tells it to find this customer. The Proxy Object asks the customer home if such an object exists. The customer home must turn to the CICS application to find the customer. If the customer is there, the Customer Object is retuned with all the customer information from the CICS application set. The proxy updates its customer information with that of the Customer Business Object. Finally, the client updates the user interface with the information in the proxy.

### 8.3.2 A Complete Workflow Scenario

In the previous section, we completed one action in detail. The other actions accessible from the user interface are very similar to the Create Customer Scenario.

Even though the application we demonstrate is far from complete, we choose a set of actions that can make up a natural workflow. This is to make the demonstration as close to a real-life situation as possible.

After the scenario steps, we provide a list of which CBConnector services we have applied.

John Smith has just finished law school and has obtained a good job in a law firm. He wants to open a checking account in the CBConnector Bank and will deposit his first pay check to this account. Since his mother, Jane Smith, loaned him $200 to buy a suit for his job interview, he wants to pay this money back to her private checking account. This is no problem because she is a customer of the same bank.

By curious coincidence, Jane Smith calls on the phone to report that her purse has just been stolen; so she needs to freeze all her accounts. The clerk does this at once, but informs her that her son is just about to deposit an

amount to her private checking account, and in this bank, it is not possible to deposit money to a frozen account. Jane Smith remembers that she only had the family checking account credit card in her stolen purse, and she agrees to unfreeze her private checking account. The teller looks up the account key for the private checking account, unfreezes this account, and the transfer can be made.

These are the steps you need to perform to accomplish this workflow. Remember that the actions may be performed in parallel.

#### 8.3.2.1 Preparation

1. Under **customer -> create customer** Jane Smith.
2. Under **checking account -> create checking account** for Jane Smith, called *private.*
3. Under **checking account -> create checking account** for Jane Smith, called *family.*

#### 8.3.2.2 Live

4. Under **customer -> create customer** John Smith.
5. Under **checking account -> create checking account** for John Smith. *Make sure to keep the generated account key.*
6. Under **checking account -> deposit** $1500 to Johns Smith's checking account.
7. *Jane Smith calls....*
8. Under **customer -> query customer** Jane Smith.
9. Under **customer -> freeze all** Jane Smith's accounts.
10. Under **customer -> query all customer's accounts** for Jane Smith. *Make a note of the private checking account key.*
11. Under **checking account -> unfreeze** Jane Smith's private checking account.
12. Under **checking account -> transfer** $200 from John Smith's checking account to Jane Smith's savings account.

#### 8.3.2.3 Clean-Up

When you have run the demo once, you need to do the steps below, and skip the preparation steps.

13. Under **customer -> unfreeze all accounts** for Jane Smith.
14. Under **customer -> remove customer** John Smith.

So which CBConnector services did we use to accomplish this demo? The following is a list of which services were used and when they were used. The services are written in italics. For an explanation of the different services, look to the *IBM Component Broker Overview,* SG24-2022, redbook and to the product manuals.

The entities in the application, such as **Customer** and **CheckingAccount**, are *Business Objects* running in an *Application* on a *Component Broker Server*. The customer's addresses are *Composed Business Objects*.

All Business Objects have a Java implementation, whereas the *specialized homes* are implemented using C++.

When we create a new customer, we make use of a *specialized home* and a CICS Procedural Application Adaptor to access the legacy tier.

Similarly, when we create a Checking account, we use the DB2 Relational Database Application Adaptor to access DB2 on the NT server.

When we list accounts for a customer, we do so by using the Query Service.

# Part 3. Appendices

Part 3 provides a description of how to install your test Ping test applications.

# Appendix A. Test the Component Broker Connector Installation

Component Broker Connector is a framework with many components. Before you start installing the CBConnector Bank, it is a good idea to make sure that your Component Broker environment is installed and functioning correctly.

There are two small test applications developed for this purpose. They are described in detail in the *IBM CBConnector Cookbook Collection: First Steps,* SG24-2033*,* redbook, and a short summary is presented here.

The applications include CBConnector server and client programs, which will verify, by running a very simple Ping application, that your client can access the CBConnector server.

You will find them on the CD-ROM in the following subdirectories:

- *CBBANK\TESTCBC\04-Transient-Ping*
- *CBBANK\TESTCBC\04-Persistent-Ping*

One of the above server applications supports only transient Managed Objects. The other server application uses the DB2 Application Adaptor to store the essential state of data in the DB2 database. Unfortunately, it is not possible to install a CICS Procedural Application Adaptor, since it always needs site-specific modifications.

You will see that there are three client implementations for the samples, but we will only test the Java client, since the sample bank application has a Java client.

These test applications use all the important basic components of Component Broker Connector, such as:

- Object Request Broker
- Naming Service
- LifeCycle Service
- Transaction Service
- Server Runtime
- DB2 Application Adaptor
- System Management

## A.1 Setting Up the Environment

Component Broker has a multitier, distributed infrastructure; so you can execute both the server and client applications directly on the CBConnector

server, or execute them separately on two different computers. We would like to advise you that you typically install and run your Ping applications first on your newly installed CBConnector server. Afterwards, install the Ping test clients on the CBconnector client machine and test the remote environment.

To install the test Ping application server and client programs, prepare your environment in the following way:

- Install Component Broker Runtime and CBConnector client on your CBConnector server and client as described in the *Component Broker Connector Quick - Beginnings Guide*.
- Install the DB2 server with the development toolkit.
- Rename the existing *somojor.zip* file, and copy the *somojor.zip* from the *04-Transient-Ping\Client\Java\Class* subdirectory to your Java CLASSPATH. Include the current path (`.\`) in the CLASSPATH.
- If you are using more that one computer, assure that there is a TCP/IP protocol properly configured between them.
- Install Location Objects by following these steps:

  1. Make sure the Name Server is running. Use the System Manager user Interface to:
     - Expand the Host Images folder; then expand the folder for your host.
     - Expand the Server Images folder; then click **Name Server**.
     - Look at the bottom of the window for the current status of the Name Server in the status line and ensure it is running.
  2. If the Name Server isn't running, restart it by following these steps:
     - From the System Manager User Interface home view, expand Management Zones, Sample Cell and WorkgroupZone, and Configuration.
     - Activate the Sample Configuration.
  3. From the CD-ROM's *Utility* subdirectory, execute the following command

     ```
     <drive>:\CBConnector\Utility\CreateAllLoc.cmd
     ```

     During the execution, you should see the following line several times:
     `Scope successfully created.`

## A.2 Install the Ping Transient Server

You are now ready to install the server. This involves four steps:

- Install the server application
- Configure the application server
- Activate the configuration and start the server

## A.3 Install the Server Application

Install the server application by running:

*<drive>:\CBConnector\04-Transient-Ping\Install\setup.exe*

Accept all the default information during the setup procedure.

### A.3.1 Configure the Application Server

Perform the following steps to configure the application server:

1. Start the System Management User Interface, and set the user level to `superuser`.
2. Create a new **Management Zone** called `Ping TR Management Zone`.
3. Create a new configuration for the newly created Management Zone called `Ping TR Configuration`.
4. Drag the **PingTRAppFam** application onto the new configuration. PingTRAppFam is found under **Host Images -> Application Family installs**. Select to **drag** it, and go back to the configuration. Select the configuration, and select **add application**.
5. On the **Ping TR Configuration**, configure a new **free-standing serve**r called `PingTRServer`. This is a predefined name, so you must use exactly this name for the server.
6. Configure the server with the host by dragging the server onto the host. Select the server, select **drag**, select your host in the host folder, and select **configure server (free standing)**.

## A.4 Activate the Configuration and Start the Server

Finally, activate the configuration by selecting the **Ping TR Configuration**, and select to **activate** it.

When the activation has completed, the System Management status line should show that the server is `running excellent`.

In case you want to stop and restart your server, do as follows:

Find your server image under **Host Images -> your host name -> server images**, and select **Stop Immediate**.

To restart, select `Run Immediate` on you server image.

## A.5 Install the Ping Transient Java Client Program

To install the Java client application, follow these steps.

From the CD-ROM, run
*<drive>:\CBConnector\04-Transient-Ping\Install\setup.exe*

Accept the default destination directory, and select the **PingJavaClient** as an install option.

## A.6 Run the Ping Transient Java Client Program

To run the Java client application, follow these steps:

1. Open a command shell window.
2. Check that there is a reference to the local path in the CLASSPATH, and that the *somojor.zip* and *jdk1.1.5/lib/classes.zip* are in the CLASSPATH.
3. Change to the directory *<drive>:\CBroker\ntApps\PingTRAppFam\bin.*
4. Enter the following command to run the Ping test client application:

   ```
   Java PingTRJ <PingId> <number of pings> <hostname.domain.company.com>
   <port number>
   ```

   For example:

   ```
   Ping TRJ 2100 cbcsrva.itsc.austin.ibm.com 900
   ```

5. The application log should look like this:

```
*** Start of Java transient Ping Client Program for CBC v. 1.2 ***
1) About to call ORB.init passing Bootstrap information as a property list:
ORBInitialHost = The host name of the initial CBConnector server
ORBInitialPort = The bootstrapping port number to connect to

2) Now attempt to resolve to root naming context using
IExtendedNaming.orb.resolve_initial_references("NameService")
NamingContextHelper.narrow(obj)

3) About to resolve the Factory Finder.
iExtendedNC.resolve_with_string
```

```
("host/resources/factory-finders/PingTRScope")FactoryFinderHelper.narrow(o
bj)
FactoryFinder resolved

4) Now attempt to get a ping Home object.
factoryFinder.find_factory_from_string("Ping.object interface"
IHomeHelper.narrow(obj)
5) Now attempt to create a PingKey object.
About to create a PingKey
Primary key created successfully
PingKey created

6) About to find ping object with pingHome.findByPrimaryKeyString
ping object found with pingHome.findByPrimaryKeyString
7) Now executing the ping method 100 times

8) Now calculate the average response time and set the ping attributes
number of pings: 100 average response time: 6 ms
Program ran SUCCESSFULLY

************ End of Sample Program ***********
```

Once the application has finished running, a new transient Ping Object is instantiated in memory with values for its attribute's `pingId`, `counter` and `avgResponseTime`.

## A.7 Install the Persistent Ping Application Server

In this section, we go through the additional steps that need to be performed in order to install the persistent Ping application server that uses the DB2 Application Adaptor.

Install the server and client as described in the previous sections. The server is found on the CD-ROM:

*<drive>:\CBConnector\04-Persistent-Ping\Install\setup.exe*

Select the server application, **PingDBS**, and the client application **PingDBJ.class**, and **Java**, as installation options.

### A.7.1 Create the Database

Create the database by running the following commands in a command shell window:

```
cd <drive>:\CBroker\ntApps\pingDBAppFam\bin
```

```
db2cmd makePingDB <userid> <password>
```

where `userid` and `password` must belong to a user with DBADMIN authority.

Bind your database by typing the following commands:

```
db2cmd db2 connect to PingDB user <userid> using <password>

db2 bind PingPO.bnd
```

You can cross-check the success of this operation by issuing a `SELECT * FROM Ping` statement in a DB2 command line window connected to the Ping database. There should be no entries in the Ping table yet.

### A.7.2 Configure the Ping Application Server

Configure the server as described in the previous sections with the following parameters:

**Management Zone:** `Ping DB Management Zone`.

**Configuration:** `Ping DB Configuration`

**Applications:** Under **PingDBAppFam**, drag the **PingDBS** application onto the **Ping DB Configuration**.

For applications that use the DB2 Application Adaptor, you must configure the **iDB2IMService** application as well.

Under **iDB2IMApplication**, drag the **iDB2IMService** application onto the **Ping DB Configuration**

**Server:** `PingDBServer` (Remember, this name must be the same).

After you activate the server, your server is up and running... almost. This server application needs to configure the XA Resource Manager since it uses the DB2 Application Adaptor.

Configure the XA Resource Manager by performing the following steps:

1. Stop the server, **PingDBServer**, which you find under **Host Images -> <host name> -> Server Images**.
2. Under the server, you may select **XA Resource Manager Images**, and then **PingDB**. Select **edit** on **PingDB**.
3. Select the **main** tab, and edit the **open string** attribute box. Change **PingDB** to **PingDB**, **<user ID>**, **<password>**.

4. Click on **Validate**, then on **Apply**, and finally on **OK** to save these changes.
5. You may now restart the server by selecting to **Run Immediate** on the PingDBServer.

## A.8 Install and Run the Persistent Ping Java Client

To install the Java client application, follow these steps.

From the CD-ROM, run
*<drive>:\CBConnector\04-Persistent-Ping\Install\setup.exe*

Accept the default destination directory, and select the **PingJavaClient** as an install option.

Run the application by entering the following two commands:

```
cd <drive>:\CBroker\ntApps\PingDBAppFam\bin
Java PingDBJ <PingId> <number of pings> <hostname.domain.company.com> <port number>
```

The result should be similar to the log of the transient example stored in the DB2 Ping table, with values for its attribute's `pingId`, `counter` and `avgResponseTime`. You can verify this by issuing a `SELECT * FROM Ping` statement in a DB2 command line window connected to the Ping database.

## Appendix B. Special Notices

This publication is intended to help designers and developers build applications using the CBConnector development environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by the product reference manuals.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

You can reproduce a page in this document as a transparency, if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

CICS
DB2
IBM ®
MVS
VisualAge

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

# Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 67.

- *IBM Component Broker Connector Overview*, SG24-2022
- *IBM CBConnector Cookbook Collection: First Steps,* SG24-2033
- *IBM CBConnector Cookbook Collection: CBConnector Bank Implementation,* SG24-5119

## C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

| CD-ROM Title | Subscription Number | Collection Kit Number |
|---|---|---|
| System/390 Redbooks Collection | SBOF-7201 | SK2T-2177 |
| Networking and Systems Management Redbooks Collection | SBOF-7370 | SK2T-6022 |
| Transaction Processing and Data Management Redbook | SBOF-7240 | SK2T-8038 |
| Lotus Redbooks Collection | SBOF-6899 | SK2T-8039 |
| Tivoli Redbooks Collection | SBOF-6898 | SK2T-8044 |
| AS/400 Redbooks Collection | SBOF-7270 | SK2T-2849 |
| RS/6000 Redbooks Collection (HTML, BkMgr) | SBOF-7230 | SK2T-8040 |
| RS/6000 Redbooks Collection (PostScript) | SBOF-7205 | SK2T-8041 |
| RS/6000 Redbooks Collection (PDF Format) | SBOF-8700 | SK2T-8043 |
| Application Development Redbooks Collection | SBOF-7290 | SK2T-8037 |

## C.3 Other Publications

These publications are also relevant as further information sources:

- *IBM Component Broker - Quick Beginnings,* G04L-2375 - *A*vailable with the CBConnector package

- *IBM Component Broker - Programming Guide,* G04L-2376 - Available with the CBConnector package
- *IBM Component Broker - CICS and IMS Application Adaptor Quick Beginnings,* G04L-2703 - Available with the CBConnector package
- *IBM Component Broker - System Administration Guide,* SC09-2704 - Available with CBConnector package
- *IBM Component Broker - Application Development Tools,* SC09-2705 - Available with the CBConnector package
- *IBM Component Broker - Application Programming Guide,* SC09-2708 - Available with the CBConnector package

These publications are relevant as further information sources:

- *Orbix 2, Programming Guide*
- *Orbix 2, Reference Guide*
- *The Essential Distributed Objects Survival Guide,* ISBN 0-471-12993-3
- *Client/Server Programming with JAVA and CORBA,* ISBN 0-471-16351-1
- *Understanding CORBA,* ISBN 0-13-459884-9

# How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com/`.

## How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/`

- **PUBORDER** – to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BokkManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

**Redpieces**

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

## How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** – send orders to:

| | **IBMMAIL** | **Internet** |
|---|---|---|
| In United States | usib6fpl at ibmmail | usib6fpl@ibmmail.com |
| In Canada | caibmbkz at ibmmail | lmannix@vnet.ibm.com |
| Outside North America | dkibmbsh at ibmmail | bookshop@dk.ibm.com |

- **Telephone Orders**

| | |
|---|---|
| United States (toll free) | 1-800-879-2755 |
| Canada (toll free) | 1-800-IBM-4YOU |
| Outside North America | (long distance charges apply) |
| (+45) 4810-1320 - Danish | (+45) 4810-1020 - German |
| (+45) 4810-1420 - Dutch | (+45) 4810-1620 - Italian |
| (+45) 4810-1540 - English | (+45) 4810-1270 - Norwegian |
| (+45) 4810-1670 - Finnish | (+45) 4810-1120 - Spanish |
| (+45) 4810-1220 - French | (+45) 4810-1170 - Swedish |

- **Mail Orders** – send orders to:

| | | |
|---|---|---|
| IBM Publications | IBM Publications | IBM Direct Services |
| Publications Customer Support | 144-4th Avenue, S.W. | Sortemosevej 21 |
| P.O. Box 29570 | Calgary, Alberta T2P 3N5 | DK-3450 Allerød |
| Raleigh, NC 27626-0570 | Canada | Denmark |
| USA | | |

- **Fax** – send orders to:

| | |
|---|---|
| United States (toll free) | 1-800-445-9269 |
| Canada | 1-800-267-4455 |
| Outside North America | (+45) 48 14 2207 (long distance charge) |

- **1-800-IBM-4FAX (United States)** or **(+1) 408 256 5422 (Outside USA)** – ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **On the World Wide Web**

| | |
|---|---|
| Redbooks Web Site | http://www.redbooks.ibm.com |
| IBM Direct Publications Catalog | http://www.elink.ibmlink.ibm.com/pbl/pbl |

**Redpieces**

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

## IBM Redbook Order Form

**Please send me the following:**

| Title | Order Number | Quantity |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | |
|---|---|---|
| First name | Last name | |
| Company | | |
| Address | | |
| City | Postal code | Country |
| Telephone number | Telefax number | VAT number |

☐ Invoice to customer number _____

☐ Credit card number _____

| | | |
|---|---|---|
| Credit card expiration date | Card issued to | Signature |

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

# Glossary

**Abstract Interface.** An abstract interface is one that is introduced in order to specify required behaviors without providing an actual implementation for them. The implementation must be provided by a derived interface. Often, the derived interface achieves this by delegating responsibilities to another object.

**Access Identity.** The identity of a principal used to specify access policies pertaining to that principal.

**Access Policies.** The rules that define whether a principal should be allowed to perform a particular operation on a particular object.

**Administrative Interface.** The interface of an object that defines its administrative and systems management behavior. Typically, the administrative interfaces of an object are only used by Systems Management and administration programs.

**Application Access Policy.** The mechanisms built into an application to control access to resources that it contains. Application access policies are enforced within the application implementation, although possibly with the assistance of security services for acquiring principal credentials. (See also object invocation access policy.)

**Application Adaptor (AA).** Provides a home for, and a certain quality of service to, its Managed Objects. It is responsible for providing systems capabilities such as identity, caching, persistence, recoverability, concurrency, and security to its Managed Objects.

**Application Adapter MixIn.** A special object provided to a Business Object by an Application Adaptor. The MixIn Object provides an implementation of various interfaces that are inherited in the CBConnector server environment.

**Application Object.** An Application Object is an object that implements the transient and persistent state of actively executing applications.

**Attribute.** An identifiable association between an object and a value. An attribute, A, is made visible to clients as a pair of operations: getA() and setA(). Read-only attributes only generate a get operation.

**Audit Identity.** The identity of a principal used to audit that principal's actions in the security system. Typically, a principal's audit identity is anonymous to the principal.

**Authentication.** The process of assuring that a principal is who they say they are; they are authentic. There are numerous ways for performing authentication which generally depend on one or more of something the principal knows (such as a secret or password), something the principal has (a badge or door key), or something the principal is (biometrics, a signature, finger-print, voice-print, retina-pattern, and so forth).

**Basic Business Object.** Single entity Business Object containing business methods. The logic and state is intended for use within business applications.

**Basic Object Adaptor (BOA).** BOA is a component of the ORB and exists on each CBConnector server. Its main function is to analyze each request received by the ORB and dispatch it to the object implementation that is the target of the request.

**Behavior.** The observable effects of an object performing the requested operation, including its results binding. See language binding, dynamic invocation, static invocation, or method resolution for alternatives.

**Bind Policy.** Bind policy determines which server of a CBConnector server group should be selected to receive the next request for a specific object. Workload Management determines which bind policies apply to a particular object according to definitions done by the CBConnector system administrator.

**Business Object.** An object containing business methods (logic) and state that is

intended for use within business applications. Business Objects are Managed Objects. In some contexts, the term Business Object in this book is used to refer to a Business Object class. It may also be used to refer to a composition of Business Object classes.

**Computer-Aided Software Engineering (CASE).** CASE refers to the use of computerized tools for the collection and transformation of application domain information necessary during software construction. Some CASE tools emphasize the front-end of the development lifecycle. These are referred to as Upper CASE tools, as distinguished from Lower CASE tools which emphasize activities near the end of the development lifecycle.

**Common Data Representation (CDR).** Low-level data representation in the General Inter-ORB Protocol (GIOP). The language representation is marshaled and demarshaled to and from the CDR format for transmission over a wire by the ORB.

**Cell Directory Service (CDS).** A component of DCE that provides the ability to assign a set of attributes to a name structured into a directory hierarchy. The CDS is used primarily within DCE to store RPC bindings, but its use is not limited to this.

**Client.** The code or process that invokes an operation on an object.

**Collection.** A logical grouping of elements. In the context of this book, each element is an object.

**Common Data Model (CDM).** The Common Data Model is a template of the CBConnector configuration data structure.

**Common Data Store (CDS).** The Common Data Store is a mechanism for storing structured data. The data in it is arranged as an arbitrarily complex tree of named objects each of which can have any number of named values.

**Composed Business Object.** Consists of multiple basic Business Objects.

**Compound Name.** A sequence of simple names that represents a traversal path through a Name Tree relative to some starting point. (See also simple name.)

**Container.** A component of an Application Adaptor that provides physical and administrative boundaries for Managed Objects. For example, a container holds Managed Objects and defines policies for them.

**Common Object Request Broker Architecture (CORBA).** CORBA is an architectural standard proposed by Object Management Group (OMG), an industry standards organization, for creating object descriptions that are portable among programming languages and execution platforms.

**CORBAservices.** The CORBAservices specify the standard interfaces of the OMG object services.

**Credentials.** Information stored in a Security Context about a principal. The information is related to a session established between a principal and a CBConnector server.

**Data Object.** An object that provides an object-oriented rendering of application and data. The data typically comes from data stores such as relational databases or CICS.

**Data Type.** A categorization of values operation arguments, typically covering both behavior and representation (such as the traditional non-object-oriented programming language notion of type).

**DII Dynamic Invocation Interface.** provides a dynamic interface to the ORB. With the help of the Interface Repository, a client can determine the interface at run time and dynamically invoke a method on it.

**Domain.** As a concept important to interoperability, a domain is a distinct scope, within which common characteristics are exhibited, common rules observed, and over which a distribution transparency is preserved.

**DSI Dynamic Skeleton Interface.** Allows a client to invoke a method on an object it had no knowledge of at compile time.

**Dynamic Invocation.** Constructing and issuing a request whose signature is possibly not known until run time.

**Dynamic Skeleton.** An interface-independent type of skeleton. It is used by CBConnector. servers to handle requests whose signatures are not known until run time.

**Embedded Aggregation.** A form of aggregation where the sub-objects remain visible from outside of the aggregate.

**Extended Naming Context (ENC).** A specialization of a naming context with extensions for properties, query, identity, administration, and name strings.

**Externalized Object Reference.** An object reference expressed as an ORB-specific string. Suitable for storage in files or other external media.

**Framework.** Ted Lewis, et al., define a framework as "an object-oriented class hierarchy plus a built-in model which defines how the objects derived from the hierarchy interact with one another."

**Home.** The logical owner of a Managed Object. A Managed Object has only one home. A home has factory and collection interfaces.

**Interface Definition Language (IDL).** IDL is a contractual, neutral, and declarative language that specifies an object's boundaries and its interfaces. IDL provides operating system and programming language independent interfaces to all services and components that reside on a CORBA bus.

**IIOP.** Internet Inter-ORB Protocol is an industry standard protocol. It defines how General Inter-ORB Protocol (GIOP) messages are exchanged over a TCP/IP network. The IIOP makes it possible to use the Internet itself as a backbone ORB through which other ORBs can bridge.

**Implementation Interface.** An interface introduced as a derivative of the most specialized interface of the object. The implementation interface is intended to designate the type of a specific implementation–the Type ID (see CORBA specification) of the implementation interface can be used within CORBA to differentiate implementations of the same interface. The implementation interface is also used to bring together the operational interface with the specific administrative interface relevant to that particular implementation.

**Implementation.** A definition that provides the information needed to create an object and allow the object to participate in providing an appropriate set of services. An implementation typically includes a description of the data structure used to represent the core state associated with an object, as well as definitions of the methods that access that data structure. It will also typically include information about the intended interface of the object.

**Implementation Definition Language.** A notation for describing implementations. The implementation definition language is currently beyond the scope of the ORB standard. It may contain vendor-specific and adaptor-specific notations.

**Implementation Inheritance.** The construction of an implementation by incremental modification of other implementations. The ORB does not provide implementation inheritance. Implementation inheritance may be provided by higher-level tools.

**Implementation Object.** An object that serves as an implementation definition. Implementation objects reside in an implementation repository.

**Implementation Repository**. A storage place for object implementation information.

**Inheritance.** The construction of a definition by incremental modification of other definitions. See interface and implementation inheritance.

**Instance.** An object is an instance of an interface if it provides the operations, signatures and semantics specified by that interface. An object is an instance of an implementation if its behavior is provided by that implementation.

**Instance Manager.** See Application Adaptor.

**Instance Manager MixIn.** See Application Adaptor MixIn.

**Interface.Proxy Object**. A listing of the operations and attributes that an object provides. This includes the signatures of the operations and the types of the attributes. An interface definition ideally includes the semantics as well. An object satisfies an interface if it can be specified as the target object in each potential request described by the interface.

**Interface Inheritance.** The construction of an interface by incremental modification of other interfaces. The IDL language provides interface inheritance.

**Interface Object.** An object that serves to describe an interface. Interface objects reside in an Interface Repository.

**Interface Repository.** A storage place for interface information.

**Interface Type.** A type satisfied by any object that satisfies a particular interface.

**Interoperability.** The ability for two or more ORBs to cooperate to deliver requests to the proper object. Interoperating ORBs appear to a client to be a single ORB.

**Interoperable Object Reference (IOR).** An IOR keeps information about the type and key of an object and the communications profiles needed to contact the CBConnector server and locate the object.

**Junction.** A junction represents a transition in a Name Tree federation between different implementations of a naming context. If a DB-based naming context is bound into a CDS-based naming context, that binding forms a junction.

**Language Binding or Mapping.** The conventions by which a programmer writing in a specific programming language accesses ORB capabilities.

**Lock.** A lock is used to coordinate concurrent use of a resource.

**Managed Object.** An object that is managed by an Application Adaptor. Managed Objects can have a complex composition of inheritance and containment relationships.

**Master Container.** The container owned by the Root Application Adaptor.

**Metadata.** Metadata is the self-descriptive information that can describe both services and information. With metadata, new services can be added to a system and discovered at run time.

**Method.** An implementation of an operation. Code that may be executed to perform a requested service. Methods associated with an object may be structured into one or more programs.

**Method Resolution.** The selection of the method to perform a requested operation.

**MixIn Object.** See Application Adaptor MixIn.

**Multiple Inheritance.** The construction of a definition by incremental modification of more than one other definition.

**Naming Context.** A naming context is a container of name bindings that associates a human-readable name to an object reference. Naming contexts support the CosNaming::Naming Context interface. (See also ENC.)

**Object.** A combination of state and a set of methods that explicitly embodies an abstraction characterized by the behavior of relevant requests. An object is an instance of an implementation and an interface. An object models a real-world entity, and it is implemented as a computational entity that encapsulates state and operations (internally implemented as data and methods) and responds to requestor services.

**Object Adaptor.** The ORB component which provides object reference, activation, and state-related services to an object implementation. There may be different adaptors provided for different kinds of implementations.

**Object Builder.** This is an application development tool that helps users develop Business Objects in CBConnector. It provides a set of SmartGuides to help users define their Business Objects, and it will generate all the necessary definition and implementation files in IDL, C++ and/or Java.

**Object Creation.** An event that causes the existence of an object that is distinct from every other object.

**Object Destruction.** Object destruction is a physical deletion of an object from main memory and permanent storage.

**Object Invocation Access Policy.** The mechanisms within the systems that transparently control access to objects. The object invocation access policy is enforced by the system without application involvement. An event that causes an object to cease to exist.

**Object Reference.** A value that unambiguously identifies an object. Object references are never reused to identify another object.

**Object Services.** IBM's CORBAservices implementation and enhancements. These include Naming, Security, LifeCycle, Event, Externalization, Identity, Query, Transaction, and Concurrency services.

**Object Management Group (OMG).** OMG is a consortium of vendors with the mission to define standards pertaining to object-oriented, distributed systems. The OMG is responsible for defining CORBA, CORBAservices, and CORBAfacilities in accordance with the Object Management Architecture.

**One-way Request.** A request where the client does not wait for completion of the request, nor does it intend to accept results. Contrast with deferred synchronous request and synchronous request.

**Operation.** A service that can be requested. An operation has an associated signature, which may restrict which actual parameters are valid.

**Operational Interface.** The interface of an object that defines its operational behavior. Typically, the operational interface of an object is used by general business applications - also known as the API (application programming interface). (See also Administrative Interface.)

**Object Request Broker (ORB).** ORB provides the means by which clients make and receive requests and responses.

**ORB Core.** The ORB component that moves a request from a client to the appropriate adaptor for the target object.

**Open Software Foundation (OSF).** OSF is a consortium of vendors who have collaboratively produced a reference implementation of the Distributed Computing Environment (DCE), along with several other de-facto standards, such as Motif.

**Object Transaction Service (OTS).** OTS is the CORBA services specification for managing atomic units-of-work over a series of method requests on recoverable objects.

**Parameter passing Mode.** Describes the direction of information flow for a operation parameter. The parameter passing modes are IN, OUT, and INOUT.

**Persistent Object.** An object that can survive the process or thread that created it. A persistent object exists until it is explicitly deleted.

**Principal.** A principal is a human user or system entity that is identified (through a security name) to the security system. Principals can be authenticated by the security system.

**Privilege Attribute.** Security information associated with a principal that can be used to decide what that principal can access.

**Propagation.** A function of the Transaction Service that transfers the transactional context of a client along with a method request to a served object. The Transaction Service supports both implicit and explicit propagation of a transaction context. Implicit propagation will occur automatically as the result of invoking any method on an object whose class inherits from CosTransactions::Transactional. Explicit propagation will occur only when the client obtains a context object and passes that as an explicit argument on a method request.

**Recoverable Object.** An object whose data is effected by committing or rolling back a transaction.

**Proxy.** The Proxy Object has the same interface as the Server Object it represents. Instead of

having the actual method implementation, its methods communicate with the ORB.

**Recoverable Server.** A server containing one or more recoverable objects.

**Referential Integrity.** The property ensuring that an object reference that exists in the state associated with an object reliably identifies a single object.

**Repository.** See Interface Repository and Implementation Repository.

**Request.** A client issues a request to cause a service to be performed. A request consists of an operation and zero or more actual parameters.

**Results.** The information returned to the client, which may include values as well as status information indicating that exceptional conditions were raised in attempting to perform the requested service.

**Restart Daemon.** A restart facility provided by the Transaction Service that may be used to automatically restart transactional processes.

**Restart repository.** A file holding information about the transactional programs being run on a machine. It is used by the restart daemon during restart of failed transactional processes.

**Root Application Adaptor.** The Root Application Adaptor provides a container that contains other Application Adaptor containers. The Root Application Adaptor Container is a bootstrap mechanism integrated directly into the CBConnector server environment

**Security Name.** The identity of a principal used to authenticate that principal to the security system. A set of security attributes are associated with a principal through the principal's security name, including the principal's access identity, audit identity, privileges, and so on. The security name is often referred to as a user ID.

**Server.** A process implementing one or more operations on one or more objects.

**Server Object.** An object that responds to a request for a service. A given object may be a client for some requests and a server for other requests.

**Service Context.** The Service Context is the information that flows from the client to the server (or from the server back to the client). The information is used to inform the server of the context running on the client; for any service, the appropriate behavior on the server is defined by the context of the client.

**Service's Context.** Services often have their own context. Service's context is information that is used to control the behavior of the service–that is, the "environment" in which the service executes.

**Signature.** Defines the parameters of a given operation including their number order, data types, and passing mode, the results if any, and the possible outcomes (normal vs. exceptional) that might occur.

**Simple Name.** A name in a naming context that identifies a particular name binding. A simple name is normally composed of an ID field and a kind field. Also referred to as a name component. (See also compound name.)

**Single Inheritance.** The construction of a definition by incremental modification of one definition. Contrast with multiple inheritance.

**Skeleton.** The object-interface-specific ORB component that assists an Object Adaptor in passing requests to particular methods.

**System Management Application (SMAPPL).** SMAPPL is a central point in which definitional configuration data is held. It also contains a copy of the Common Data Model (CDM). Only one host houses the central point in a management network topology.

**SMI.** Systems Management Interface. An interface supported by an object that supports operations that allow the object to be managed by a systems management service. Typically, the systems management interface is introduced to object services in the administrable interface.

**State.** The time-varying properties of an object that affect that object's behavior.

**Static Invocation.** Constructing a request at compile time. Calling an operation through a stub procedure.

**Stub.** A local procedure corresponding to a single operation that invokes that operation when called.

**Synchronous Request.** A request where the client pauses to wait for completion of the request. Contrast with deferred synchronous request and one-way request.

**Transactional Object**. An object whose behavior is affected by being invoked within the scope of a transaction.

**Transactional Server.** A server containing one or more transactional objects.

**Transient Object.** An object whose existence is limited by the lifetime of the process or thread that created it.

**UUID.** Universally Unique Identifier - A value constructed with an algorithm that provides a reasonable assurance that the identity value is unique within the known universe. Typically, UUIDs are 16 bytes long.

**Value.** Any entity that may be a possible actual parameter in a request. Values that serve to identify objects are called object references. Workload Management enhances the ORB by allowing management of Business Object instances across CBConnector servers. It minimizes client complexity by having a single system image and single object image for objects across groups of CBConnector servers.

## List of Abbreviations

| | |
|---|---|
| AA | Application Adaptor |
| AIX | Advanced Interactive Executive |
| AMS | Application Management Specification |
| API | Application Programming Interface |
| APPC | Advanced Program-To-Program Communication |
| BO | Business Object |
| BMS | Basic Mapping Support |
| BOA | Basic Object Adaptor |
| BOIM | Business Object Instance Manager alias for BOIM Application Adaptor |
| CASE | Computer-Aided Software Engineering |
| CBConnector/SM | CBConnector Systems Management |
| CDM | Common Data Model |
| CDS | Common Data Store |
| CDR | Common Data Representation |
| CICON | CICS/IMS Connection |
| CICS | Customer Information Control System |
| CLI | Call Level Interface |
| COM | Component Object Model |
| CORBA | Common Object Request Broker Architecture |
| CRUD | Create, Retrieve, Update, and Delete |
| DAO | Data Access Object |
| DB2 | Database 2 |
| DCE | Distributed Computing Environment |
| DCOM | Distributed Component Object Model |
| DII | Dynamic Invocation Interface |
| DLL | Dynamic Link Library |
| DO | Data Object |
| DSI | Dynamic Skeleton Interface |
| DSOM | Distributed System Object Model |
| DTD | Document Type Description (part of the XML standard) |
| DTS | Direct-to-SOM |
| ECD | Edit, Compile, Debug |
| ECI | External Call Interface |
| ESIOP | Environment Specific Inter-ORB Protocols |
| GIOP | General Inter-ORB Protocol |
| HOD | IBM Host on Demand |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IDE | Integrated Development Environment |
| IDL | Interface Definition Language |
| IIOP | Internet Inter-ORB Protocol |

| | |
|---|---|
| IM | Instance Manager alias for Application Adaptor |
| IMS | Information Management System |
| IOM | Interlanguage Object Model |
| IOR | Interoperable Object Reference |
| IR | Interface Repository |
| ISV | Independent Software Vendor |
| JIT | Just-in-Time |
| MDL | Model Definition Language |
| LU 6.2 | Logical Unit Type 6.2 |
| MFS | Message Format Services |
| MO | Managed Object |
| MQSeries | Message Queuing Series |
| ODBC | Open Database Connectivity |
| OLE | Object Linking and Embedding |
| OLTP | On-line Transaction Processing |
| OO-SQL | Object Oriented-Structured Query Language |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| PAA | Procedural Application Adaptor |
| PAO | Procedural Adaptor Object |
| PDA | Personal Digital Assistant |
| QOP | Quality Of Protection |
| RACF | Resource Access and Control Facility |
| RAS | Reliability, Availability, Serviceability |
| RDBMS | Relational Database Management System |
| RPC | Remote Procedure Call |
| RRBC | Release-to-Release Binary Compatibility |
| SAO | Server Administration Object |
| SLI | Single Logical Image |
| SMAPPL | Systems Management Application |
| SOM | System Object Model |
| SPI | System Programming Interface |
| TME | Tivoli Management Environment |
| TO | Transaction Object |
| TR | Transaction Record |
| UML | Unified Modeling Language |
| UUID | Universally Unique Identifier |
| VSAM | Virtual Sequential Access Method |
| WLM | Workload Management Enhanced Client |
| .XML | Extended Markup Language |

# Index

## S



## T



## U



## V



## W



## X

# ITSO Redbook Evaluation

IBM CBConnector Cookbook Collection CBConnector Bank User Guide
SG24-5121-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ **Customer** _ **Business Partner** _ **Independent Software Vendor** _ **IBM employee**
_ **None of the above**

**Please rate your overall satisfaction** with this book using the scale:
**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

Overall Satisfaction _____

**Please answer the following questions:**

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

**Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

SG24-5121-00
Printed in the U.S.A.