

CA-IDMS[®] Server

User Guide

4.2



Computer Associates

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED, OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

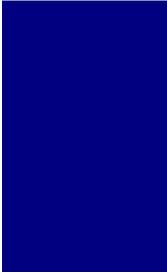
The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

First Edition, December 1999

© 1999 Computer Associates International, Inc.
One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.



Contents

Chapter 1: Introducing CA-IDMS Server

Overview	1-1
About This Guide	1-3

Chapter 2: Preparing to Install CA-IDMS Server

Overview	2-1
Hardware and Software Prerequisites	2-1
Installing CA-IDMS Server	2-4
Setting Up CA-IDMS Server	2-4
Setting Up CAICCI/PC	2-5

Chapter 3: Setting Up Your Mainframe Environment

Overview	3-1
Installing CA-IDMS Server	3-1
Setting Up CA90s Services	3-1
Defining the CA-IDMS System	3-2
Setting Up Database Access	3-4

Chapter 4: Installing CA-IDMS Server on the PC

Overview	4-1
Before Installing CA-IDMS Server	4-1
Installing CA-IDMS Server	4-2

Chapter 5: Managing Data Sources

Overview	5-1
Defining Data Sources	5-2

Setting Up a Server	5-7
Setting ODBC Options	5-9
Specifying CA-IDMS Administrator Defaults	5-13
Logging Errors and Trace Information	5-14
Setting Language Options	5-15

Chapter 6: Using the CA-IDMS Java Server

Overview	6-1
Installing the Java Server	6-1
Configuring the Web Server	6-2
Configuring the Java Server on Windows NT	6-2
Using the Java Server on Windows NT	6-3
Using the Java Server on Other Platforms	6-4

Chapter 7: Connecting to a Data Source Using ODBC

Overview	7-1
Connecting to a Predefined Data Source	7-2
Connecting Dynamically to a Data Source Not Previously Defined	7-3

Appendix A: ODBC Programmer Reference

Overview	A-1
Debugging User Sessions	A-1
ODBC Conformance Levels	A-2
Database Type Mapping Between ODBC and CA-IDMS	A-8
SQLDriverConnect Connection String Format	A-10
Driver-Specific Connect Options	A-11
Supported Isolation and Lock Levels	A-12
Bulk Insert Support	A-12

Appendix B: JDBC Programmer Reference

Overview	B-1
JDBC Conformance	B-1
Database Type Mapping Between JDBC and CA-IDMS	B-3
Connection Parameters	B-6

Appendix C: Registry Information

Overview	C-1
Registry Information	C-1

Appendix D: Passing Accounting Information to CA-IDMS

Overview	D-1
Supplying Accounting Information	D-1

Appendix E: Configuring CAICCI for TCP/IP

Overview	E-1
Specifying Protocol Parameters for TCP/IP	E-2
Tracing a Communications Problem	E-4
Testing the Configuration	E-5
Exiting the CAICCI Configurator	E-6

Index

Introducing CA-IDMS Server

Overview

CA-IDMS Server provides open database access to data stored in CA-IDMS databases. Open database access allows users to maintain existing corporate databases and make the data available to new Windows or browser applications. CA-IDMS Server provides support for dynamic SQL using both the ODBC and JDBC application program interfaces.

CA-IDMS Server 4.2 supports ODBC on Windows 95, Windows 98, Windows NT, and Windows 2000 and supports JDBC on all platforms that support Java 1.1.

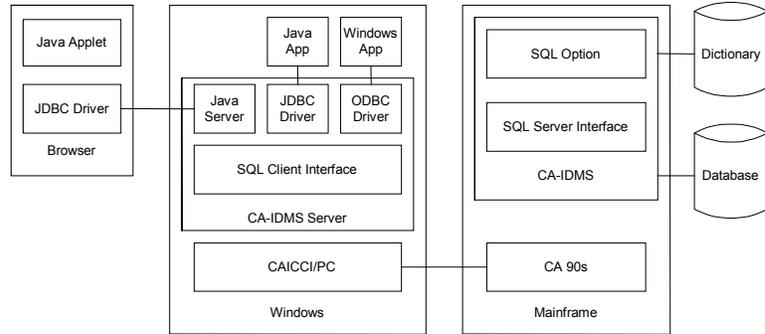
CA-IDMS Server 3.0 runs on Windows 3.1 and provides support for ODBC applications as well as compiled SQL embedded in CA-Visual Realia COBOL applications. CA-IDMS Server 3.0 must be used in the Windows 3.1 environment.

Who Should Use this Document

You should use this document if you are a CA-IDMS database administrator, a CA-IDMS system administrator, an ODBC or JDBC application developer, or an end user of an application using CA-IDMS Server to access a CA-IDMS database. If you are a database administrator or end user, use this document to define data sources to access CA-IDMS data from a client application. System administrators should use this document to set up a CA-IDMS system for access by the CA-IDMS Server. If you are an application developer, use this document to understand how ODBC and JDBC API requests are implemented in CA-IDMS Server. This document assumes that you are familiar with the Windows user interface and can perform basic Windows and mouse operations.

Components of CA-IDMS Server

CA-IDMS Server 4.2 consists of ODBC and JDBC drivers, a Java Server that supports JDBC applets running in Java enabled browsers, and the native CA-IDMS SQL client-server interface. It uses the CA90s Services to communicate to the mainframe using either the TCP/IP or LU2 communications protocols. The following diagram shows how the CA-IDMS Server software components fit together:



Delivery of CA Components

Delivery of the CA software components utilized by CA-IDMS Server 4.2 is shown in the chart below.

Software Component	Base Product	Installation Medium
ODBC Driver Manager	Microsoft Data Access Components	CA-IDMS Server 4.2 CD or Microsoft Web Site
CA-IDMS Server ODBC Driver	CA-IDMS Server	CA-IDMS Server 4.2 CD
CA-IDMS Server JDBC Driver	CA-IDMS Server	CA-IDMS Server 4.2 CD
CAICCI/PC	CA90s Services	CA90s Services ¹ tape or CA-IDMS Server 4.2 CD
CAICCI	CA90s Services	CA90s Services ² tape
CAIENF	CA90s Services	CA90s Services ² tape
CA-IDMS	CA-IDMS	CA-IDMS
CA-IDMS Server (server interface)	CA-IDMS Server	CA Host Server or CA-IDMS ³

¹ For BS2000: Located on the CA-IDMS Server 4.2 CD.

² For BS2000: Located on the CA-IDMS tape.

³ Please consult the product cover letter for the appropriate tape for your environment.

About This Guide

This guide assumes you are a new user of CA-IDMS Server and that you have experience using Microsoft Windows. This guide also assumes you have familiarity with CA-IDMS and database access.

In addition to this chapter, this guide includes:

- Chapter 2, “Preparing to Install CA-IDMS Server,” which provides a summary of the prerequisites and steps for installing all of the product components.

- Chapter 3, “Setting Up Your Mainframe Environment,” which describes information you need to define on the mainframe to establish communications between CA-IDMS and your Windows application using CA-IDMS Server.
- Chapter 4, “Installing CA-IDMS Server on the PC,” which describes how to install CA-IDMS Server.
- Chapter 5, “Managing Data Sources,” which explains how to use the Microsoft ODBC Administrator application to define CA-IDMS data sources.
- Chapter 6, “Connecting to a Data Source,” which explains how to use the CA-IDMS DriverConnect dialog boxes to connect to a data source.
- Chapter 7, “Using the CA-IDMS Server,” which explains how to configure and run the Java Server service that allows Java applets to access CA-IDMS databases.
- Appendix A, “ODBC Programmer Reference,” which describes ODBC conformance levels supported by CA-IDMS Server, data type mapping between ODBC and CA-IDMS, and driver-specific option numbers.
- Appendix B, “JDBC Programmer Reference,” which describes JDBC conformance supported by CA-IDMS Server and data type mapping between JDBC and CA-IDMS.
- Appendix C, “Registry Information,” which describes the information stored in the registry used by CA-IDMS Server 4.2.
- Appendix D, “Passing Accounting Information to CA-IDMS,” which discusses how to pass accounting information from the PC client to the backend CA-IDMS system using CA-IDMS Server.
- Appendix E, “Configuring CAICCI/PC for TCP/IP” contains information about using CAICCI options.

For More Information

CA-IDMS Server
Online Help

CA-IDMS Server provides a context-sensitive online reference to dialog boxes in the form of online Help. To get online Help about a dialog box, click the Help button displayed in the dialog box.

Related
Documentation

You may need to refer to the following documents when setting up CA-IDMS to work with the ODBC server:

- *CA-IDMS Installation*
- *CA-IDMS System Generation*
- *CA-IDMS System Operations*
- *CA-IDMS Database Administration*
- *CA-IDMS SQL Reference*

- *CA-IDMS SQL Programming*
- *CA-IDMS DML Reference – COBOL*
- *CA Host Server Installation*
- *CA90s Services documentation, especially the CA90s Services CAICCI User Guide*

Refer to Microsoft documentation and support for more information about Microsoft ODBC.

Preparing to Install CA-IDMS Server

Overview

For CA-IDMS Server to function, its components must be installed on both the client workstation and the mainframe. Additional prerequisites are a properly configured CA-IDMS environment on the mainframe and properly configured communications interfaces on both the mainframe and the client.

This chapter summarizes the following requirements for running CA-IDMS Server:

- PC hardware prerequisites
- PC software prerequisites
- Mainframe software prerequisites
- CA-IDMS Server installation
- CA-IDMS Server setup

Hardware and Software Prerequisites

CA-IDMS Server requires the mainframe software and the PC hardware and software described in this section.

PC Hardware Prerequisites

To install and use CA-IDMS Server on your PC, you need the following hardware:

- Communications hardware as described in CA90s Services CAICCI documentation
- Graphics card compatible with Microsoft Windows
- Hard disk with at least 4 MB free space on your hard drive for CA-IDMS Server

PC Software Prerequisites

CA-IDMS Server requires the following software to be installed on the PC:

- Microsoft Windows 95, Windows 98, or Windows NT 4.0 (Service Pack Release 3 recommended) or higher
- TCP/IP or LU2 communications protocols
- CAICCI/PC
- ODBC Driver Manager 2.5, or later
- Java 1.1, or later virtual, machine for JDBC support

About CAICCI/PC

CAICCI/PC is a CA90s Services component that provides a common interface between the CA-IDMS ODBC Driver and various communications protocols. CAICCI/PC is distributed on the CA90s Services tape for your mainframe operating system. For your convenience, CAICCI/PC is included on the CA-IDMS Server 4.2 installation CD (ccipcw32.exe in the CCI directory). Note that this is a self-extracting exe file and must be copied to a writable disk before it can be run.

About the ODBC Driver Manager

The ODBC Driver Manager provides the link between ODBC enabled applications and ODBC drivers and is installed automatically by newer releases of many Microsoft products, including Windows NT, Office, and Internet Explorer. CA-IDMS Server 4.2 conforms to the ODBC 2.5 specification and will work with the ODBC 2.5, or later, Driver Manager. You can verify the version of ODBC installed on your machine clicking on the ODBC 32 icon in the Windows Control Panel and selecting the About tab.

Microsoft also distributes the ODBC Driver Manager as part of the Microsoft Data Access Components (MDAC). The latest version of MDAC can be downloaded from Microsoft's web site, www.microsoft.com. For your convenience, MDAC 2.0, which includes the ODBC 3.5 Driver Manager, is included on the CA-IDMS Server 4.2 installation CD (mdac_min.exe in the Microsoft directory).

About the Java Virtual Machine

A Java Virtual Machine (JVM) is an interpreter that executes Java programs, which are stored on disk as "class" files. CA-IDMS Server 4.2 conforms to the JDBC 1.2 specification and requires a Java 1.1 compliant Virtual Machine. Recommended Java Virtual Machines for CA-IDMS Server 4.2 are the JVM installed as part of the Microsoft Internet Explorer 4.01, Service Pack 1, or later, and the Sun Microsystems Java Run Time Environment (JRE) 1.1.8 or later.

The latest version of the Microsoft Java Virtual Machine can be downloaded from Microsoft's web site, www.microsoft.com. For your convenience, the Microsoft VM for Java Version 5.0 is included on the CA-IDMS Server 4.2 installation CD (MSJavx86.exe in the Microsoft directory).

You can verify the version of the Microsoft Java virtual machine, if installed, by entering "jview" at the command prompt. The commands used to verify the version of the SUN Java virtual machine vary depending on the version and whether the JRE or JDC is installed, but are generally either "jre" or "java-version".

The latest version of the Sun Microsystems Java Run Time Environment can be downloaded from Sun's web site, www.java.sun.com. For your convenience, the JRE 1.1.8 and 1.2 for Windows are included on the CA-IDMS Server 4.2 installation CD (jre1_1_8-win.exe and jre12-win32.exe in the Sun directory).

For More Information

Refer to the following documentation:

- For information about installing CA90s Services on the mainframe, downloading and configuring CAICCI/PC, and supported network transport products, CA90s Services documentation.
- For additional information about configuring CAICCI/PC for LU2 using the CAICCI/PC Configurator for Windows NT and Windows 95, *CA90s Services CAICCI User Guide*.
- For additional information on CAICCI/PC for TCP/IP see Appendix E.
- Additional information about ODBC and MDAC is available on the World Wide Web at www.microsoft.com.
- Additional information about JDBC and Java is available on the World Wide Web at www.java.sun.com.
- Please read Appendix A for detailed information regarding CA-IDMS Server 4.2 conformance to the ODBC standard.
- Please read Appendix B for detailed information regarding CA-IDMS Server conformance to the JDBC standard.

Mainframe Software Prerequisites

CA-IDMS Software CA-IDMS Server requires that CA-IDMS Release 12.0 (or higher) be installed on the mainframe. You also need the SQL option.

Refer to the *CA-IDMS Installation and Maintenance Guide* for your mainframe operating system for information about installing CA-IDMS on the mainframe.

CA90s Services Software CA-IDMS Server requires that the following CA90s Services software be installed on the mainframe:

- CAIENF
- CAICCI

For information about installing CAIENF and CAICCI on the mainframe, refer to CA90s Services documentation.

Installing CA-IDMS Server

To install CA-IDMS Server, you perform an install procedure on:

- The mainframe where CA-IDMS is installed
- Each workstation PC that will access CA-IDMS databases on the mainframe

For Step-By-Step Instructions

Refer to the following documentation:

- To install CA-IDMS Server on the mainframe, *CA-IDMS Installation*
- To install CA-IDMS Server on the PC, Chapter 4, "Installing CA-IDMS Server on the PC," in this manual

Setting Up CA-IDMS Server

Mainframe To set up CA-IDMS Server access on the mainframe, each CA-IDMS system to be accessed as a CA-IDMS data source server must be generated with the following definitions:

- A CCI line
- A PTERM/LTERM pair for each concurrent session with the CA-IDMS system
- A CA-IDMS Server task
- The SQL definitions in the catalog area of the dictionary associated with the CA-IDMS system

PC	To set up CA-IDMS data source definitions on the PC, you must run the ODBC Administrator application.
For Step-By-Step Instructions	Refer to the following chapters in this document: <ul style="list-style-type: none">■ To set up CA-IDMS Server on the mainframe, Chapter 3, “Setting Up Your Mainframe Environment”■ To set up CA-IDMS Server on the PC, Chapter 5, “Managing Data Sources”

Setting Up CAICCI/PC

You can use the CCI Configurator to configure CAICCI for all data sources.

When CAICCI/PC is configured for TCP/IP, you can use the CA-IDMS ODBC administrator to configure CAICCI/PC differently for different data sources.

Refer to “Setting Up a Server” in Chapter 5 for information on setting CCI options for a data source.

For detailed information on how to configure CAI/PC, refer to the *CA90s Services CAICCI User Guide*. Appendix E also contains information about configuring CAICCI/PC.

Setting Up Your Mainframe Environment

Overview

This chapter describes what you need to do on the mainframe to establish communications between your Windows application and CA-IDMS using CA-IDMS Server. The discussion describes:

- Installing CA-IDMS Server
- CA90s Services requirements
- CA-IDMS system-generation parameters necessary to establish communications
- Things to consider when you define a database to be accessed by CA-IDMS Server

Installing CA-IDMS Server

CA-IDMS Server must be installed on the mainframe. This host component of CA-IDMS Server is delivered on the CA-IDMS installation tape for Release 14.0 and higher, and on the CA-Host Server tape for prior releases. Refer to *CA-IDMS Installation and Maintenance* or *CA Host Server Installation* for more information about installing CA-IDMS Server on the mainframe.

Setting Up CA90s Services

CA-IDMS Server uses the CAIENF (Event Notification Facility) and the CAICCI (Common Communications Interface) components of CA90s Services.

For information about CAIENF, CAICCI, and supported communications protocols refer to CA90s Services documentation.

Defining the CA-IDMS System

The following information describes how to define the CA-IDMS system-generation parameters to support communications using CA-IDMS Server. Include these definitions in each CA-IDMS system to be accessed from a CA-IDMS data source.

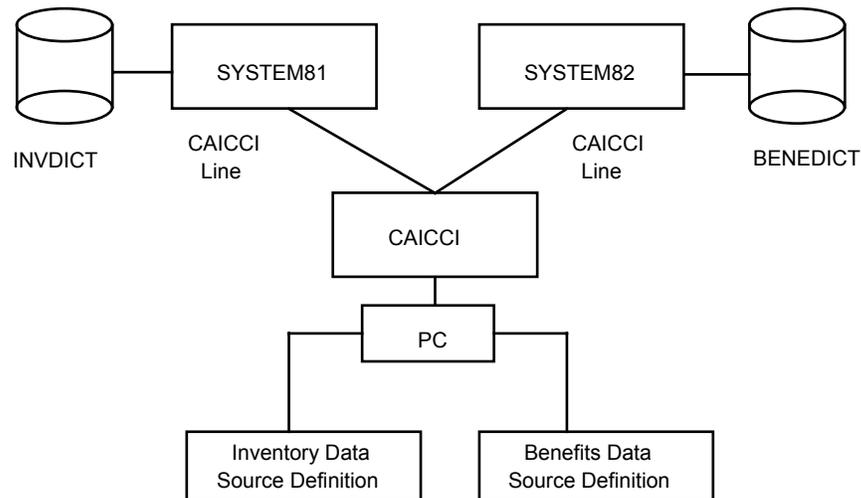
For detailed information about these system-generation statements and how to use the system-generation compiler, refer to *CA-IDMS System Generation*. Additionally, *CA-IDMS System Operations* provides information about configuring and maintaining CA-IDMS systems.

Examples

The examples in this chapter assume that CA-IDMS Server will be used to access CA-IDMS systems, System 81 and System 82, from a PC.

Associated with System 81 is INVDICT, an Inventory dictionary. Associated with System 82 is BENEDICT, a Benefits dictionary. INVDICT and BENEDICT contain the definitions of the tables to be accessed from the PC. "Data source" definitions on the PC refer to the mainframe dictionary names.

The following diagram illustrates the CA-IDMS system network:



Defining a CAICCI Line

CAICCI provides communication between mainframes or between a mainframe and PC that is independent of any particular communications protocol or access method. A CAICCI line connects a CA-IDMS system with the CAICCI network. Define a CAICCI line in each CA-IDMS system that will be accessed as a CA-IDMS data source.

Add one CAICCI line for the CA-IDMS system you are defining plus a PTERM and LTERM pair for each concurrent session with the CA-IDMS system.

By default, each Windows process uses one PTERM/LTERM pair when connected to a CA-IDMS system. Define a minimum of one PTERM/LTERM pair for each PC that will be used concurrently to access the CA-IDMS system. If more than one application will be used on the same PC to access the CA-IDMS system, you must define additional PTERM/LTERM pairs for each additional application. For example, if 5 PC users each run 2 ODBC applications concurrently, define 10 PTERM/LTERM pairs.

When multi-threading is enabled each ODBC or JDBC connection uses one PTERM/LTERM pair. For example, if you have a single PC running a World Wide Web server application, and wish to support concurrent access to CA-IDMS by 25 World Wide Web browser clients, define 25 PTERM/LTERM pairs.

Example

This example defines a CAICCI line for System 81. The LINE, PTERM, and LTERM statements below define a CAICCI line with two physical terminals, which allows two PC users the ability to be logged on to System 81 at the same time:

```
ADD SYSTEM 81
    SYSTEM ID IS SYST0081

ADD LINE CCILINE
    TYPE IS CCI.

ADD PTERM PTECCI01
    TYPE IS BULK

ADD LTERM LTECCI01
    PTERM IS PTECCI01.

ADD PTERM PTECCI02
    TYPE IS BULK

ADD LTERM LTECCI02
    PTERM IS PTECCI02.
```

Creating the CASERVER Task

The TASK statement defines a task and its characteristics, including the code used to invoke the task. The default task code is CASERVER. You can override the default task code if you want to control resources per user or to apply additional security.

The CASERVER task code is similar to the RHDCNP3S task code, which controls the resource limits and time-out values for CA-IDMS external user sessions. Define a CASERVER task code for each CA-IDMS system that will be accessed by CA-IDMS Server. To do this:

1. From the system-generation compiler, enter this command:

display task rhdcnp3s as syntax.

The system-generation compiler displays the definition of the RHDCNP3S task code.

2. Erase the DISPLAY TASK statement at the top of the screen.
3. Change the name of the task code, RHDCNP3S, to CASERVER (or whatever task code you have chosen) and modify the task definition like this:
 - Add the INTERNAL parameter if it is not already there.
 - Make the value for the EXTERNAL WAIT parameter appropriate for your users; for example, 1800 (half an hour).

Note: If you do not define a CASERVER task code, CA-IDMS uses the RHDCNP3S task code to define the characteristics for a CA-IDMS Server session.

Setting Up Database Access

CA-IDMS Server uses dynamic SQL to access a CA-IDMS database from an ODBC or JDBC application. Both the SQL Option and the mainframe component of CA-IDMS Server must be installed on the CA-IDMS system. The database can be defined using the Schema compiler or SQL DDL. In either case, you must include the appropriate SQL definitions in the dictionary associated with the CA-IDMS system. The SQL definitions reside in the catalog area of the dictionary.

Setting Up SQL Access

The following recommendations are useful to set up SQL access to CA-IDMS databases:

- To access a non-SQL-defined database using SQL, you must define an SQL schema that identifies the network schema and the segment where the data is stored. The network schema must conform to the rules described later in this chapter.
- If the application does not qualify table references with schema names define one or more CA-IDMS/DC profiles that set a default schema name.
- To limit the size of the list of tables returned to an ODBC application, you may create an "accessible tables" view that returns a subset of the the default view and set it as an option for specific data source. Using such a view of accessible tables may generate a more meaningful list of tables for each user and improve performance. For more information about accessible tables views, see the "Setting ODBC Options" topic in the CA-IDMS Server 4.2 ODBC Administrator Help. Note that since registry settings are generally not available to Java applets, the JDBC driver always uses the default view, "SYSCA.ACCESSIBLE_TABLES".

- Define views in the catalog to provide easy access to non-SQL-defined databases or application-specific data. For example, consider using a view when tables are to be joined using the set-name condition. Remember, however, that views created by joining two or more tables cannot be updated.
- Implement table procedures to provide easy access to non-SQL-defined databases or application-specific data. For example, consider using a table procedure to navigate a complicated network database. Table procedures can also be used to update databases.

Utilizing Page Groups

A page group is a physical database definition attribute set by the DBA during database definition. The catalog and the target database can be in different page groups. Unless the CA-IDMS 14.1 Mixed Page Group Feature is activated, tables from mixed page groups cannot be accessed with a single request. Additionally, once a table from one page group has been accessed, a COMMIT must be issued before a table from a different page group can be accessed without an intervening COMMIT.

Review these tips to use data sources defining data from mixed page groups:

- If the catalog contains definitions of tables from mixed page groups, define a different data source and a different Accessible Tables View for each page group. Each Accessible Tables View should include tables from a single page group. Therefore, the end user will not be allowed to accidentally access mixed page groups after a table list function is performed.
- Use the Automatic Commit option when accessing tables in different page groups.

Mixed page groups are supported by CA-IDMS Release 14.1, and these restrictions do not apply where the data source is on a 14.1 system and the Mixed Page Group Binds feature has been activated.

Note: Automatic Commit is enabled by default and can be disabled using an ODBC or JDBC function.

See CA-IDMS Database Administration — Volume 1 or the CA-IDMS 14.1 Features Guide for more information about using page groups.

Setting Up SQL Access to Non-SQL Databases

This section reviews the transformations used by the SQL engine when it reads definitions of non-SQL records. When you use SQL to access non-SQL records, the entity names you code in the SQL syntax must follow the conventions described below.

Accessing Non-SQL Records Using SQL Statements

To reference an SQL table in SQL statements, you code the table name preceded by a schema name qualifier. For example, in this statement:

```
SELECT * FROM DEMOSCH.SAMPLE
```

SAMPLE is the table name and DEMOSCH is the SQL schema in which it is defined. The combination of schema name and table name allows the SQL compiler to look up the definition of the table in the SQL catalog.

To access a non-SQL record from an SQL statement, you must code the record name the same way. To do this, define an SQL schema that maps to the corresponding non-SQL schema. Then, use the SQL schema name to qualify all subsequent references to non-SQL records in SQL DML statements. For example:

```
CREATE SCHEMA SQLNET
  FOR NONSQL SCHEMA PRODDICT.CUSTSCHM;
SELECT * FROM SQLNET."ORDER-REC";
```

For more information about defining SQL schemas, see the *CA-IDMS SQL Reference* for syntax and information about accessing non-SQL databases and *CA-IDMS Database Administration – Volume 1* for process-related information.

Transforming Non-SQL Record and Set Names

Non-SQL record and set names may contain embedded hyphens, which are allowed in the naming conventions for non-SQL schemas, but not in the naming conventions for SQL schemas. To use record and set names with embedded hyphens in an SQL statement, enclose the names in double quotes (for example, "CUST-REC-123").

Transforming Non-SQL Element Names

CA-IDMS automatically transforms embedded hyphens in non-SQL element names to underscores when they are referenced through SQL. Therefore, to access the CUST-NUMBER element in a non-SQL record, you must code CUST_NUMBER in an SQL statement.

Creating SQL Synonyms

When a FOR LANGUAGE SQL synonym is defined for a non-SQL record, CA-IDMS uses the element synonyms for all SQL access. SQL synonyms are used **only** for element names.

Defining SQL synonyms for non-SQL records is sometimes the only way to overcome column name limitations within SQL. Some non-SQL element names don't make satisfactory SQL column names, even after the hyphen is changed to an underscore. For example, if a non-SQL element name starts with a numeric character, you must still use double quotes around the element name; for example, to access 123-ORD-NUM, you would code "123_ORD_NUM" in an SQL statement.

Elements That Cannot Be Transformed

Group elements, REDEFINES elements, FILLERS, and OCCURS DEPENDING ON elements are simply not available for access by SQL. From the view point of the SQL user, it is as if these elements simply weren't defined in the non-SQL record. However, the subordinate elements of a group definition are available, as are the base elements to which a REDEFINES is directed.

Fixed Occurs Element Definitions

Though OCCURS ... DEPENDING ON declarations are not available for SQL access, fixed OCCURS definitions are. To the SQL user, a fixed OCCURS element appears as one column for each occurrence of the element. The column name for each occurrence is the original element name followed by an underscore and an occurrence number. If the element is declared with multiple OCCURS levels, the corresponding column names contain one underscore and one occurrence number for each "dimension" of the OCCURS declaration. For example, the element definition BUD-AMT OCCURS 12 TIMES will generate the following column names:

```
BUD_AMT_01, BUD_AMT_02, BUD_AMT_03, . . . , BUD_AMT_12.
```

Note, in the preceding example, that the occurrence number appended to the column name is large enough to hold the largest subscript from the corresponding element definition.

The base element name in combination with the appended occurrence information cannot be greater than 32 characters. If it is, you must define an SQL synonym for the non-SQL record.

Note also that, although the CA-IDMS SQL implementation allows 32-character column names, other SQL implementations restrict column names to 18 characters. In particular, some ODBC client software may require you to use SQL synonyms for non-SQL records to limit the size of the transformed column names to 18 characters.

Tip: Another way to define shorter names is to create a view of the record and specify view column names.

Defining CALC Keys, System-Owned Index Set Keys, and User-Owned Sort Set Keys

To access a control-key definition (of a CALC, INDEX, or sorted set) using SQL, the control-key definition must not include a FILLER element. If it does, change the non-SQL record definition to assign a name other than FILLER to the element(s) in question.

In addition, the control-key definition cannot incorporate the subordinate elements of a group level REDEFINES when these elements are smaller in size than the base element being redefined, as in this example:

```
02 ELEM1 PIC X(8) .
02 ELEM1REDEF REDEFINES ELEM1 .
03 ELEM1A PIC S9(8) COMP .
03 ELEM1B PIC S9(8) COMP .
```

An error occurs if ELEM1A and ELEM1B are used in the control key definition because they are smaller than the element that they redefine (even though together they are the same as ELEM1). When this condition occurs, change the redefining group, which contains the smallest subordinate elements, into the base-element definition. Use the base-element definition in the control key specification. For example, ELEM1REDEF should be the base-element definition in the sample above, and ELEM1 should be coded so that it REDEFINES ELEM1REDEF.

Setting up Catalog Views

Both ODBC and JDBC provide "metadata" APIs that return information about schemas, tables, columns, and indexes from the SQL catalog. CA-IDMS Server uses table procedures and views in the SYSCA schema to access information catalog information. In general these are installed into the catalog when CA-IDMS is installed. Additional views may need to be defined, depending on the release of CA-IDMS.

SYSCA.ODBC_INDEX

This table procedure is used to return index and CALC key information for network records as well as SQL tables. The ODBC driver uses this table procedure to implement the SQLStatistics and SQLSpecialColumns functions. If it is not installed the ODBC driver queries the catalog SYSTEM.INDEX and SYSTEM.INDEXKEY tables, and returns information only for SQL defined tables. The JDBC driver uses it to implement the corresponding DatabaseMetaData getIndexInfo and getBestRowIdentifier methods. This table procedure is required for JDBC support.

Gen level 9506 or later of CA-IDMS 12.01, or later, is required to use this feature, and includes the DDL to define the table procedure (contained in VIEWDDL) and the table procedure load module (IDMSOINX) which is installed in the same library as the updated CA-IDMS nucleus modules. In most cases the table procedure is already defined. If not, the DDL to define it is contained in IDMSOINX.DDL, which has been installed in the CA-IDMS Server directory.

SYSCA.ACCESSIBLE_
SCHEMAS

This view returns a list of schemas that contain tables that are accessible to the user. This view is used by the JDBC driver to implement the DatabaseMetaData.getSchemas method, and is required for JDBC support. It is not used by the ODBC driver.

For CA-IDMS Release 14.1 and earlier releases, the DDL contained in ACCVIEWS.DDL, installed in the CA-IDMS Server\Java directory, should be run against every catalog that will be accessed using JDBC.

For More Information

Refer to the *CA-IDMS SQL Reference* for more information about accessing non-SQL-defined databases using SQL.

Installing CA-IDMS Server on the PC

Overview

This chapter describes what you need to know to install CA-IDMS Server 4.2 on your PC. It also identifies what you need to do before you install the product.

Before Installing CA-IDMS Server

Before you install CA-IDMS Server the following must be installed:

- Network transport software, either LU2 or TCP/IP.
- CAICCI/PC
- ODBC Driver Manager
- Java Virtual Machine to use the JDBC driver

Obtain the following information from your CA-IDMS system administrator:

- The name of the dictionary that contains the definitions of the tables you want to access.
- The node name of the system on which the dictionary resides.
- The task code defined for the CA-IDMS Server. The default task code is CASERVER.

On Windows NT a user id with administrative privileges must be used when installing CA-IDMS Server and CAICCI/PC.

Installing CA-IDMS Server

1. To allow shared components to be updated properly you should exit all Windows applications before beginning the installation of CA-IDMS Server. This includes Microsoft Office tool bars.
2. Run **setup.exe** from the setup directory of the CA-IDMS Server 4.2 CD.
3. Before copying the files from the installation disk onto your system, the Installer displays the readme file, which contains information not available when this document was prepared.
4. Select Typical to install all components of CA-IDMS Server 4.2 including the ODBC driver, JDBC driver, Java Command Facility demo, and "javadoc". Select Compact to install the ODBC driver only. Select Custom to choose which components to install.

Note: The ODBC driver should always be selected, since it installs components also used by the JDBC driver.

5. The CA-IDMS Server 4.2 Installer updates the registry with information required by the ODBC Driver Manager. It then displays a dialog box that allows you to add to the registry any data sources, defaults, and options that were defined for CA-IDMS Server 3.0 16 bit data sources. You can add data sources as either user data sources, which are available to the currently signed-on user only, or as system data sources, which are available to all users and services of the machine. Server definitions, defaults, and global options are always available to all users and services.

Generally, data sources that will be used by Windows NT services should be defined as system data sources. Data sources must be defined as system data sources to be used by the CA-IDMS Java Server. See Chapter 6 for information about the CA-IDMS Java Server.

6. The CA-IDMS Server 4.2 Installer displays a dialog box that allows you to run the ODBC Administrator to modify or add additional data source, server, and option definitions.

The CA-IDMS Server 4.2 menu will be added to the Start Menu.

Use the Microsoft ODBC Administrator to define CA-IDMS data sources to be accessed through ODBC and JDBC.

Managing Data Sources

Overview

A data source is a description of a database you want to access from an ODBC or JDBC application. You use the ODBC Administrator from the Control Panel or Start menu to define a connection between the PC and CA-IDMS.

When you define a data source, you must include the names of the dictionary where the SQL tables are defined and the CA-IDMS system that contains the dictionary. You can also specify other options.

The CA-IDMS JDBC driver can use ODBC data source definitions to access CA-IDMS databases. The JDBC driver does not use the ODBC driver at run time.

This chapter discusses the elements of a data source definition and how to use the ODBC Administrator to define and administer a data source definition. The following table describes when to use each tab on the CA-IDMS ODBC Administrator property sheet.

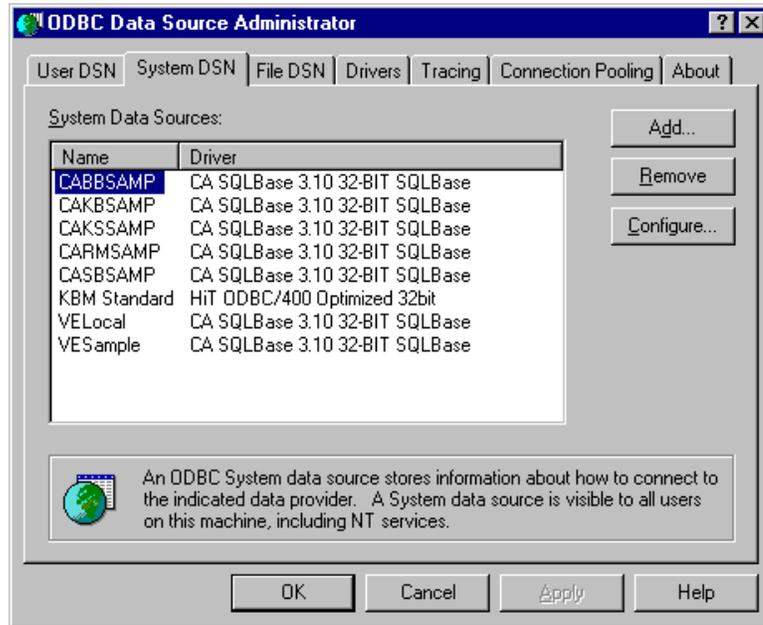
Tab	Use It When	Documented In
Data Source	You define a new data source or modify the dictionary of server or an existing data source.	Defining Data Sources
Server	The name you supply in the Server combo box is not the CA-IDMS system name <i>or</i> You want to add or change a Server definition <i>or</i> You want to override the default CCI options for this server.	Setting Up a Server

Tab	Use It When	Documented In
Options	You want to change the default runtime attributes for the data source	Setting ODBC Options
Defaults	You are likely to change a particular default ODBC option setting every time you define a CA-IDMS data source	Specifying CA-IDMS Administrator Defaults
Log Options	You want to override the default log file specification and options, or to enable/disable tracing	Logging Errors and Trace Information
International	You want to select the Country Extended Code Page or Double Byte Character Set used to translate character data transferred between PC and host	Setting Language Options
Java Server	You want to configure the CA-IDMS Java Server to allow access to CA-IDMS from browser applets.	Configuring the Java Server

Defining Data Sources

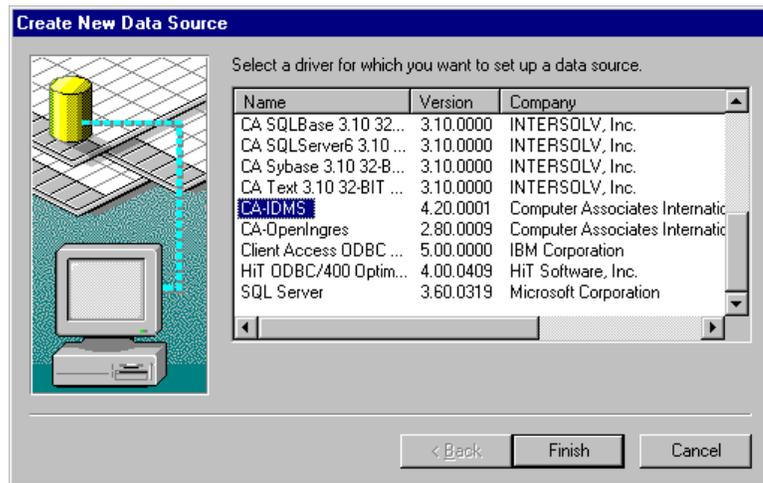
When you double-click the Microsoft ODBC Administrator icon in the Windows Control Panel, the Data Sources dialog box appears. This box lists the name of each defined data source followed by the database driver in parentheses.

If no data source definition exists, the Data Sources dialog box looks like this:

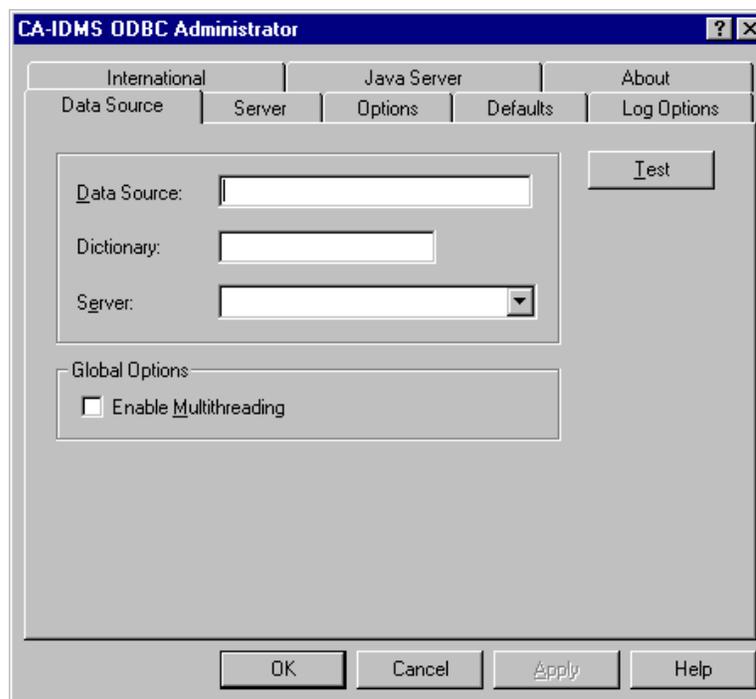


Adding a Data Source

To create a new CA-IDMS data source definition, choose the Add button. The Add Data Source dialog box appears, listing installed drivers. Select CA-IDMS.



Click the Finish button. The CA-IDMS ODBC Administrator property sheet appears.



In the Data Source tab of the CA-IDMS ODBC Administrator property sheet, you supply the essential information for a data source definition – the data source, the name of the dictionary containing definitions of the databases, and the name of the CA-IDMS server.

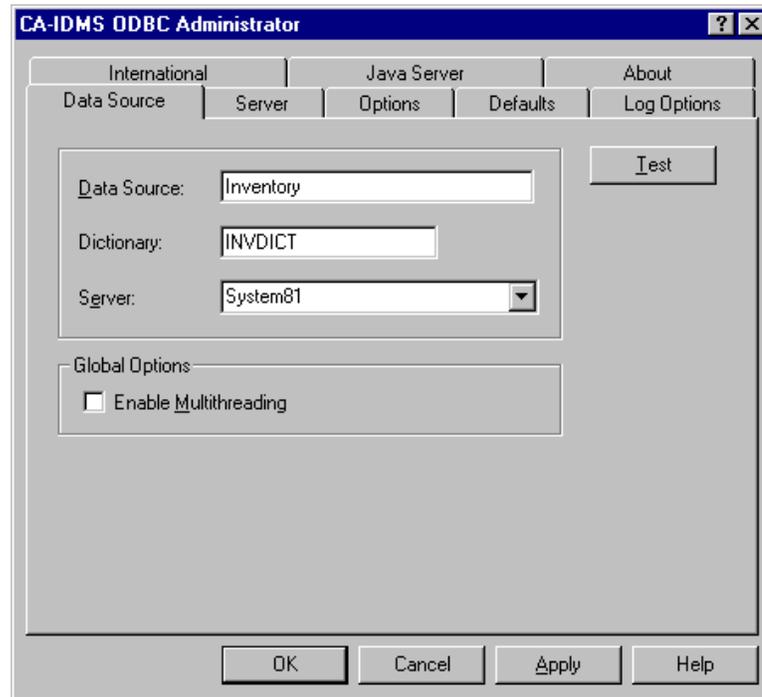
Data Source specifies a data source name. When adding a new data source, enter a character string of up to 32 characters in the Data Source text box. You can use a combination of letters, numbers, spaces, or special characters. When modifying an existing data source its name cannot be changed.

Dictionary specifies the DBNAME or segment name of the dictionary that contains the definitions of the tables you wish to access. This name must be defined in the DBNAME table on the CA-IDMS system identified by the server name. This is optional; if not entered the default is the first eight characters of the data source name.

Server specifies the name of the server that identifies the CA-IDMS node containing the dictionary defining the tables you wish to access. You can enter a new name or select an existing name from the pull down list. This is required and can be either the node name of the host CA-IDMS system or a 1-to-32 character logical name of a Server that defines the node name and options.

Enable Multithreading allows concurrent access to the CA-IDMS system by all connections in the same application. By default only one connection can communicate with a CA-IDMS system at a time. Note that this option affects all data source descriptions.

After you complete the dialog box, it looks something like this:



Note: The information displayed in this chapter is based on the sample CA-IDMS system network illustrated in Chapter 3, “Setting Up Your Mainframe Environment.” The values displayed are based on the system-generation statements defined for System 81 and System 82.

Saving the Data Source Definition

After creating and testing the data source definition in the CA-IDMS ODBC Administrator, choose OK to save the definition to the registry and return to the Data Sources dialog box. Choose Apply to save the definition without leaving the CA-IDMS ODBC Administrator. Choose Cancel to return to the Data Sources dialog box without saving the definition.

Testing the Data Source Definition

You can use the Test button to run the Test Connect application to verify that the data source is defined correctly and that CA-IDMS Server is installed correctly. It displays this dialog box.



In the **User Id** text box, supply a valid user ID for the CA-IDMS system associated with the data source named in the **Data Source** text box. If required for the system, supply the password in the **Password** text box. Then choose Connect. The test program will connect to the data source using the CA-IDMS ODBC driver.

Setting Up a Server

Use the Server tab to define a Server. A server is used to define the CA-IDMS system that contains the databases.

The screenshot shows the 'CA-IDMS ODBC Administrator' dialog box with the 'Server' tab selected. The dialog has a title bar with a question mark and a close button. Below the title bar are tabs for 'International', 'Java Server', and 'About'. Underneath are sub-tabs for 'Data Source', 'Server', 'Options', 'Defaults', and 'Log Options'. The 'Server' sub-tab is active, showing the following fields:

- IDMS Options:**
 - Name:** System81
 - Node Name:** SYST0081
 - Via Node:** (empty)
 - Task Code:** CASERVER
- CCI Options:**
 - Wait Timeout:** 0
 - Server Name:** (empty)
 - Server Port:** 0

At the bottom of the dialog are buttons for 'OK', 'Cancel', 'Apply', and 'Help'. A 'Delete' button is located to the right of the 'Name' field.

Name displays the selected Server name. This is entered on the Data Source page and cannot be changed here.

Node Name specifies the node name of the system containing the tables you wish to access as specified by the SYSTEM ID of the system generation parameters. This is optional and defaults to the first eight characters of the server name if none is specified. Note that the first eight characters of the server name must be in upper case if this is not specified.

Via Node optionally specifies the node that CCI will establish a connection with. The system identified by Via Node must contain a RESOURCE table entry for the system identified by node name. Use this option when the system containing your tables does not directly communicate with CCI.

Task Code specifies an alternate task code to be used for statistics and limit checking. If no value is entered, the default task code of CASERVER is used. The value you enter must be defined to the CA-IDMS system using the TASK system generation statement.

The multi-threaded version of CCI must be installed to use the following options, although multi-threading need not be enabled for the ODBC driver.

Wait Timeout specifies the number of seconds to wait for a reply from the server. This overrides the Reply Wait Timeout specified using the CCI Configurator for this Server definition only. When this timeout is exceeded a communications error is returned and the connection can no longer be used. If multi-threading is enabled for the application can continue processing other connections. Enter 0 to use the default value set by CCI. Entering -1 is equivalent to the largest positive integer, which essentially waits indefinitely.

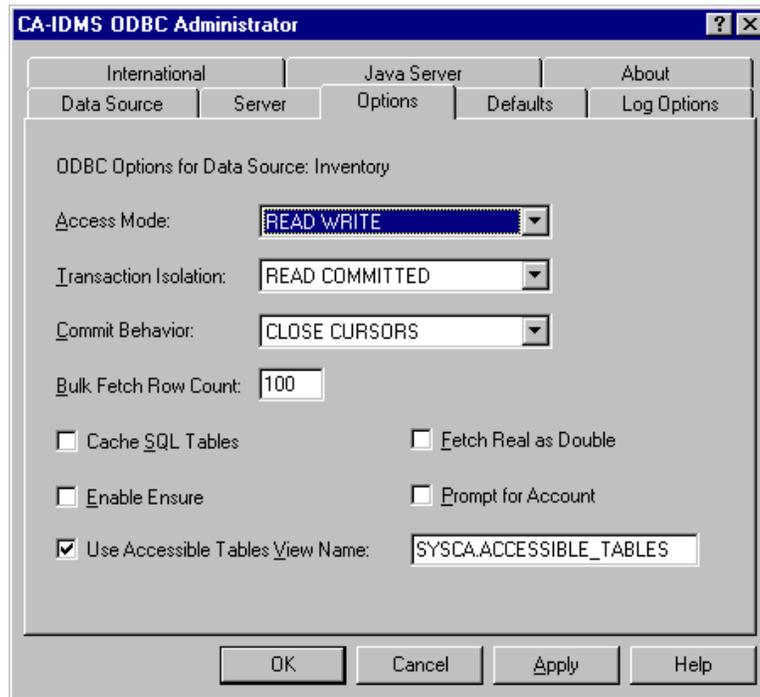
Server Name specifies the name or TCP/IP address of the CCI host server. This overrides the default CCI server name specified using the CCI Configurator for this ODBC server only, and allows concurrent access to multiple CCI servers.

Server Port specifies the TCP/IP port of the CCI host server. This overrides the default CCI server port specified using the CCI Configurator for this Server only. Enter 0 to use the default value set by CCI, which is normally 1202.

Delete deletes the server definition and closes the dialog box.

Setting ODBC Options

Select the Options tab to set ODBC options for a specified data source. These options are used only by the ODBC driver.



The data source name is protected. To modify the displayed defaults for database access using this data source, choose among the options presented in the dialog box.

Access Mode determines whether the application can both read and update the CA-IDMS database or only read it. The default, READ WRITE, allows applications to read and update the database. This option is defined by the ODBC specification and can be set and queried by the application using the `SQLSetConnectOption` and `SQLGetConnectOption` functions.

READ ONLY is recommended unless you intend to update data in the CA-IDMS database.

Transaction Isolation determines the degree to which your transactions are impacted by (and impact) other users accessing the same data. The choices are:

- READ COMMITTED, which protects you from accessing data updated by another user before it has been committed. READ COMMITTED corresponds to the CURSOR STABILITY option of the CA-IDMS SQL SET TRANSACTION statement.

- **READ UNCOMMITTED**, which provides no such protection, permits only retrieval operations to be executed by the user; update requests are rejected. This option can be selected only in conjunction with an Access Mode of **READ ONLY**. **READ UNCOMMITTED** corresponds to the **TRANSIENT READ** option on the **SET TRANSACTION** statement.

This option is defined by the ODBC specification and can be set and queried by the application using the **SQLSetConnectOption** and **SQLGetConnectOption** functions. For more information about the **SET TRANSACTION** statement, see the *CA-IDMS SQL Reference*.

Commit Behavior determines how a **COMMIT** operation affects cursors in CA-IDMS. The choices are:

- **CLOSE AND DELETE CURSORS**, which closes and deletes cursors; the application must prepare and execute the next statement.
- **CLOSE CURSORS**, which closes cursors; applications can execute a statement without calling prepare again.
- **PRESERVE CURSORS**, which preserves cursors in the same position as before the **COMMIT** operation; the application can execute or fetch without preparing the statement again.

This is a CA-IDMS Server extension that allows you to optimize the way different applications use ODBC. The ODBC application can use **SQLGetInfo** function to query this setting.

Accept the default of **CLOSE CURSORS** unless a specific application requires a different setting.

Bulk Fetch Row Count specifies how many database rows should be fetched in a single database request. The default is 100. Valid values are 1 to 30000. Bulk fetch can improve performance.

Choose the default unless experience suggests another value for performance reasons.

Cache SQL Tables causes the CA-IDMS Server to cache the table list returned by the **SQLTables** function. This is used to improve performance and can reduce the amount of time it takes to retrieve a list of tables.

Although the default setting is off, enabling this is recommended.

Enable Ensure causes the CA-IDMS Server to honor the **ENSURE** parameter of the **SQLStatistics** function call. Setting **ENSURE** on this function call would normally result in an **UPDATE STATISTICS** command being issued to CA-IDMS SQL against the named table. For large tables, this can cause deadlocks or communication timeout errors.

Accept the default which disables the ENSURE option unless a specific application requires otherwise.

Fetch Real as Double causes the CA-IDMS Server to return single precision floating point numbers as double precision. Use this option to avoid rounding that can occur in numbers when passed from the mainframe to the PC. Note that some loss of precision is unavoidable when converting between the floating point formats because different numbers of bits are used to encode the exponent and mantissa.

Prompt for Account causes the SQLDriverConnect function to display a dialog box if the optional Account parameter is not passed on the connection string. See the *CA-IDMS Server Administration* manual for more information.

Use Accessible Tables View Name causes the ODBC driver to use the SYSCA.ACCESSIBLE_TABLES view (or another view defined by you) for the SQLTables function, instead of using the catalog tables directly. This allows for tailoring the list of tables presented to the user. For example, you may want to limit tables based on schema or authorization. The SYSCA.ACCESSIBLE_TABLES view restricts access to only those tables which can be accessed by the user. This is also used when security requirements do not allow direct access to the catalog tables.

Checking the Use Accessible Tables View Name box enables an edit control in which you can enter the name of the view. The default is the SYSCA.ACCESSIBLE_TABLES view. A different view can be defined in the catalog and entered here. If you specify a different name, be sure that it contains the same columns as SYSCA.ACCESSIBLE_TABLES (although it can also contain additional columns).

For information about the SYSCA.ACCESSIBLE_TABLES view, see the *CA-IDMS SQL Reference*.

Select the OK button to save your changes to the database access defaults in the registry. Select the Cancel button to close the dialog box without saving any new changes.

Performance Considerations for ODBC Options

The options you choose for ODBC can affect the performance of CA-IDMS Server ODBC driver. The list below identifies specific options that can affect performance:

- **Cache SQL Tables** — Use this option to increase performance at the expense of not necessarily having the most current view of existing tables. When you turn this option on, CA-IDMS Server will use the cached result to process repeated SQLTable requests. CA-IDMS Server flushes the cache whenever you turn off this option, change the request parameters, change the name of the Accessible Tables view, or disconnect from a session.
- **Enable Ensure** — The ENSURE parameter of the SQLStatistics functional call in an application normally results in an UPDATE STATISTICS command against the named table. This can be a very lengthy operation that can cause your application to time out and cause contention with other users. To override the SQLStatistics function call in the application and to prevent an application from issuing a command to update table statistics, keep the Enable Ensure option off.
- **Use Accessible Tables View Name** — By default, CA-IDMS Server uses the SYSCA.ACCESSIBLE_TABLES view for the SQLTables function. When a catalog contains a large number of table definitions, you may be better off creating a tailored view of the tables that are of interest to the end user.

Specifying a view name allows the list of tables to be tailored to the user. For example, the SYSCA.ACCESSIBLE_TABLES view returns only those tables to which the user has SELECT authority. If you define your own views, the view definition must include the following columns:

```
SCHEMA (CHAR(18))  
TABLE (CHAR(18))  
TYPE (CHAR(1))
```

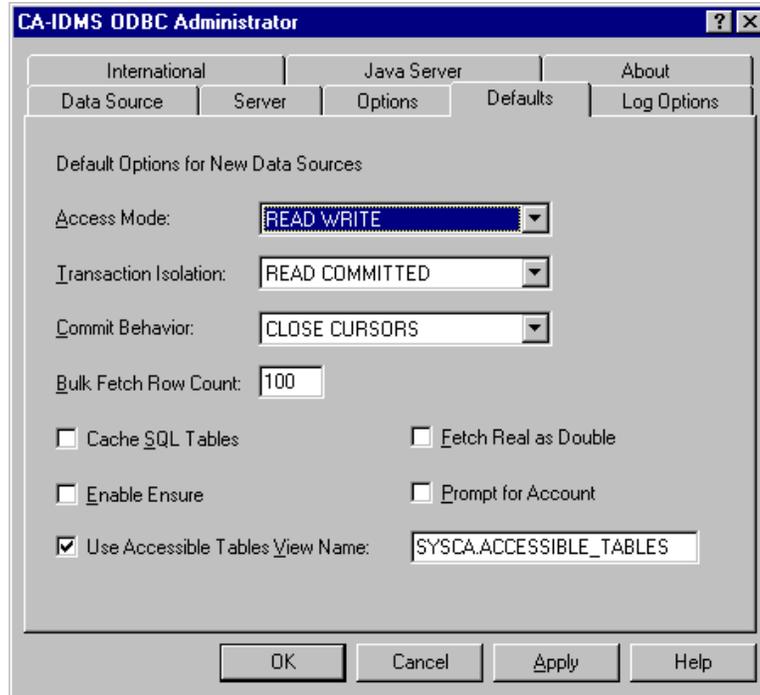
Refer to the *CA-IDMS SQL Reference* for information about the SYSCA.ACCESSIBLE_TABLES view.

Refer to Microsoft documentation about open database connectivity software for more information about the various ODBC functions mentioned in the descriptions above.

Specifying CA-IDMS Administrator Defaults

Administrator defaults establish the default option settings for a new data source. The CA-IDMS Administrator applies the settings each time you define a new data source. Existing data source definitions are not affected.

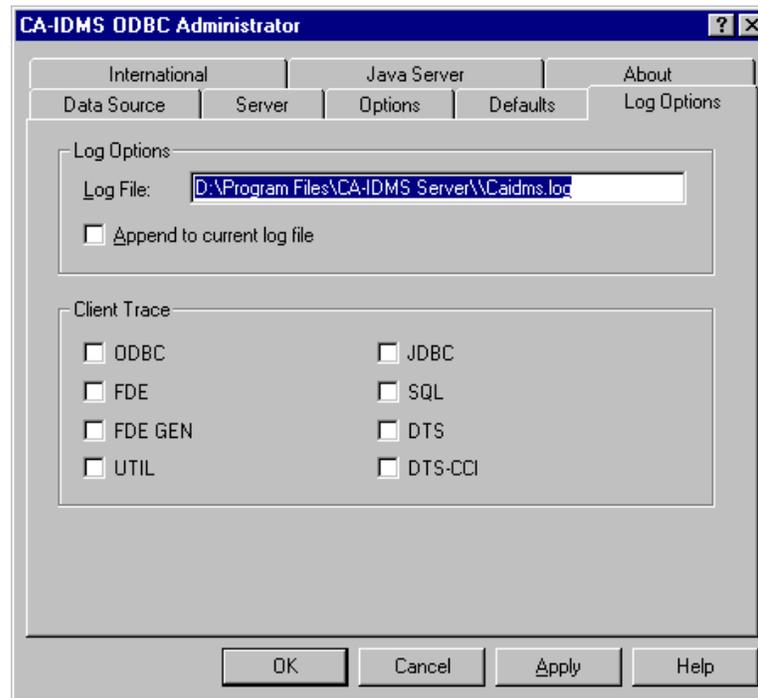
Use the Defaults tab to change administrator defaults.



For a description of each option, see ODBC options.

Logging Errors and Trace Information

CA-IDMS Server writes messages for some types of errors to a log file. You can specify the name and directory of this log file using the Log Options tab. You can also enable tracing of JDBC, ODBC, SQL, and internal function calls on this tab.



Normally, you would enable tracing only to research a problem in conjunction with technical support. Log options affect all data sources. For example, if you specify a log file name, all trace entries are written to the specified file. You cannot specify different log options for different data sources.

The default log file name is CAIDMS.LOG. CA-IDMS Server writes messages to the log file about the status of the database connection. Client trace options control tracing on the PC.

Log File Options

Log File Name specifies the name of the log file into which the CA-IDMS Server enters messages that indicate the status of the database connection. If you omit path information in the file name, the CA-IDMS Server creates the file in the directory specified during installation to contain the CA-IDMS Server files. The log file must be in a directory available for write access by all users. The log file name cannot be set or queried at run time.

Append causes the log file to be appended with new information every time an ODBC session is started. If this option is chosen, care should be taken to clean out existing file information that is no longer needed.

Client Trace Options Use these check boxes to control tracing on the client:

- JDBC** enables tracing of calls to the JDBC driver.
- ODBC** enables tracing of calls to the ODBC driver.
- SQL** enables tracing of calls to the native SQL client interface.
- DTS** enables tracing of calls to the Data Transport Services interface.
- DTS-CCI** enables tracing of calls from DTS to CCI.
- FDE** enables tracing of Format Descriptor Element data conversion calls.
- UTIL** enables tracing of internal utility calls.

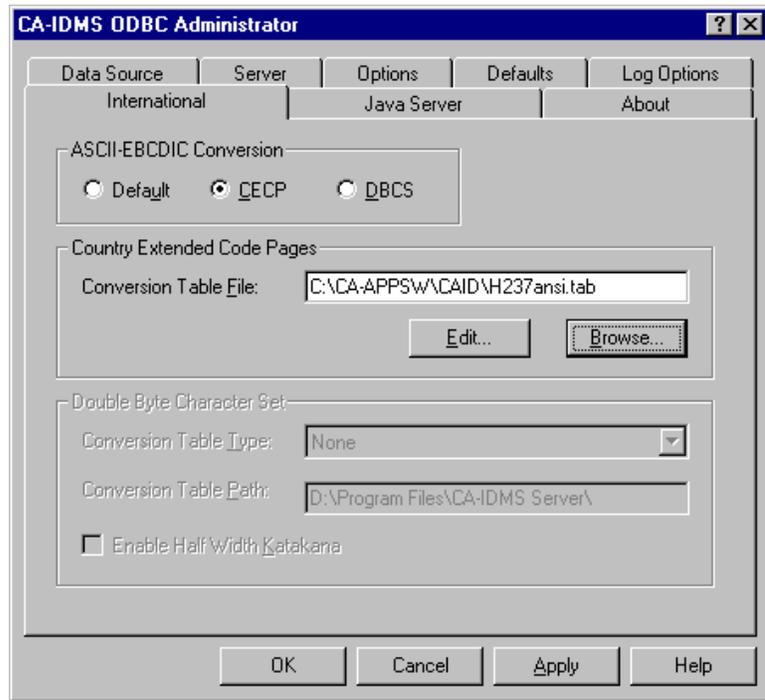
Setting Language Options

When CA-IDMS transfers character data between the host system and a PC, it uses translation tables based on English as spoken in the United States (U.S. English). You may need to override the default and create a customized translation table if your host system or PC uses code pages based on another language.

Note: The language setting is a global option which affects all data sources. You cannot establish different language options for different data sources. These conversions are performed for both the ODBC and JDBC drivers.

Selecting, Creating, and Editing CECP Translation Tables

Use the International tab to enable Country Extended Code Pages (CECP) for converting data transferred between the CA-IDMS system and the ODBC application. Click the **CECP** radio button to enable the CECP options, then select the file containing the conversion tables. Click the **Default** button to use the default conversion tables.

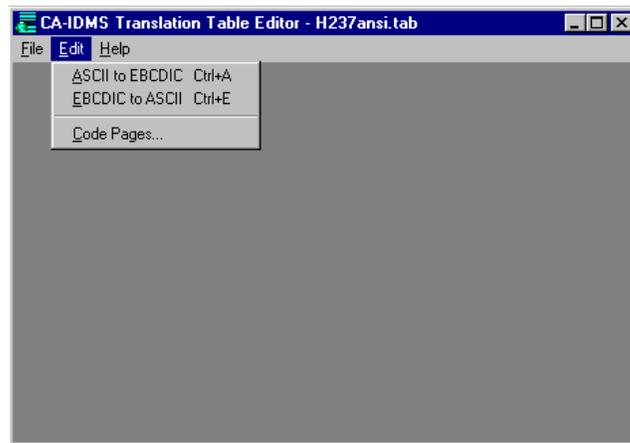


Selecting a Translation Table

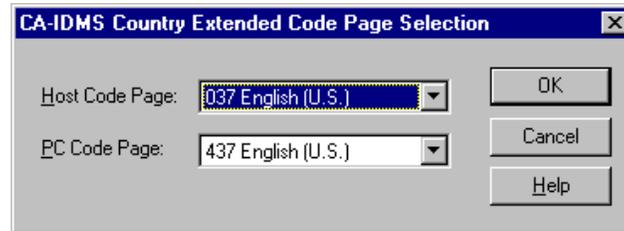
To select a translation table, enter the name of the table file in the CECP Conversion Table File Name text box or choose the Browse button to select from a list of available files.

Creating or Editing a Translation Table

Choose the Edit button to create or edit a translation table. When you select the Edit button, CA-IDMS Server activates the Translation Editor. To open an existing translation table, choose Open from the File menu. For a new or existing translation table, choose Code Pages from the Edit menu to access the CA-IDMS Country Extended Code Page Selection dialog box.



This dialog box, shown below, allows you to select the code pages to use for your translation table.



The Host Code Page list box includes these Country Extended Code Pages for the EBCDIC character set on the mainframe:

Code Page	Representative Language
037 English (U.S.)	English and most other European languages
273 German, Austrian	German and Austrian German
277 Norwegian	Norwegian
278 Finnish, Swedish	Finnish and Swedish
280 Italian	Italian
284 Spanish	Spanish
285 English (U.K.)	English and most European languages
297 French (AZERTY)	French, using the AZERTY keyboard
500 Belgian, Swiss	Belgian, Swiss French, and Swiss German

The **PC Code Page** list box includes these Country Extended Code Pages for the ASCII character set:

Code Page	Representative Language
437 English (U.S.)	English and most other European languages
850 Multilingual (Latin I)	Most languages using the Latin alphabet
852 Slavic (Latin II)	Slavic languages using the Latin alphabet
860 Portuguese	English and Portuguese
863 Canadian-French	English and French Canadian
865 Nordic	Scandinavian languages (that is, Swedish, Norwegian)

Code Page

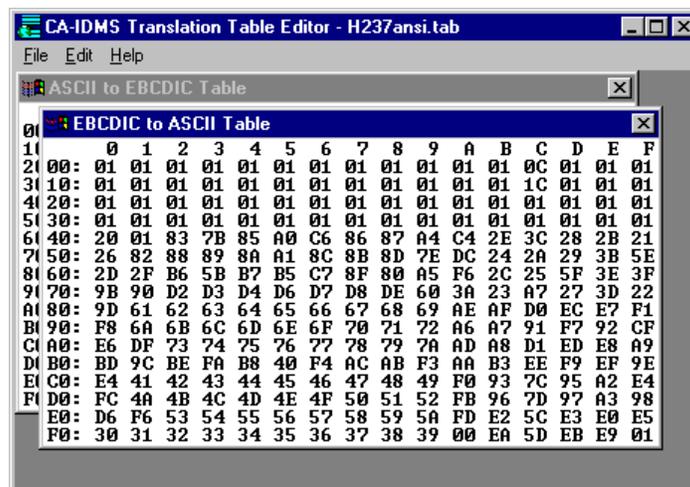
Representative Language

Customizing a Translation Table

After you create a translation table, you may need to add EBCDIC/ASCII conversions that are not supported in the standard code pages. The Translation Table Editor provides two edit windows – one for the ASCII to EBCDIC translation table and the other for the EBCDIC to ASCII translation table. To activate each window, select the appropriate option from the Edit menu.

Each window displays a table of 256 hexadecimal values. Each entry in the table represents the output character set code value indexed by the input character set code value. For example, this window represents the ASCII to EBCDIC translation table for Canadian French on the PC and U.S. English on the host machine. The ASCII value for a space (' ') in the Canadian French code page is 20 (in hexadecimal). The corresponding EBCDIC value for a space in the U.S. English code page is 40.

The generated tables convert control codes between their ASCII and EBCDIC equivalents, where possible. Versions. Note that prior to CA-IDMS Server 4.0 all control codes were converted to null bytes (x'00').



To customize the table, use the mouse or keyboard to select a hexadecimal value and replace it with another. The editor ignores all characters except the numbers 0 through 9 and letters A through F (including lowercase). You can use the mouse to move the cursor or these keystrokes:

Key	Moves Cursor
Arrow keys	One digit in the direction of the arrow
Home	To beginning of row
End	To end of row
PageUp	To top row
PageDown	To bottom row
Enter	Beginning of next row
Ctrl+Left Arrow, Right Arrow	Left or right one entry
Ctrl+Home, End	Beginning or end of table

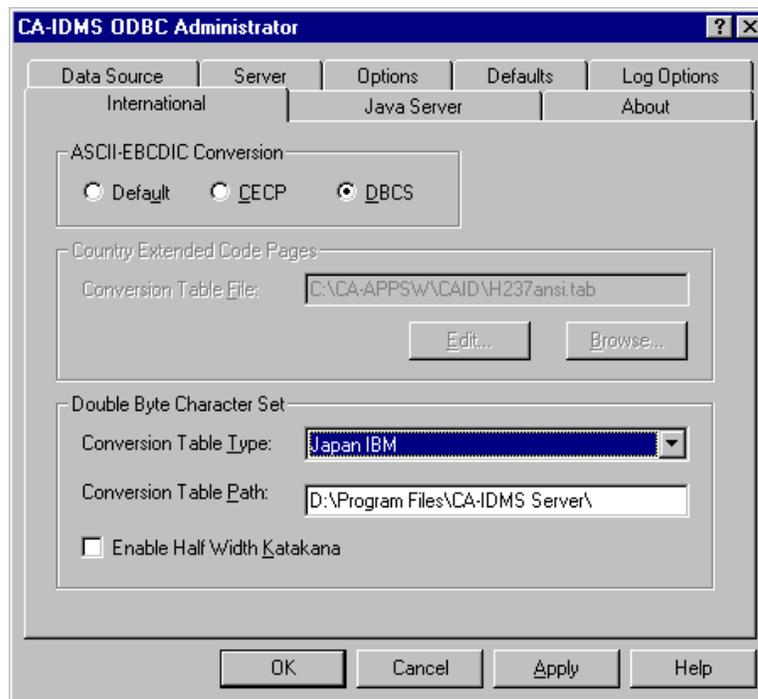
Saving a Translation Table To save a new translation table under a new name, choose Save As from the File menu. To save an existing translation table, choose Save from the File menu. The default file extension for translation table files is .TAB.

Included Tables Several code page conversion tables are provided with this release and are installed in the CA-IDMS Server directory. These files are identified by the extension "tab" and contain tables used at run time to convert between ASCII and EBCDIC. These tables are:

h237ansi.tab	Host code page 237 (Austrian/German) to ANSI. Control codes are converted from EBCDIC to x'01'. Control codes are converted from ASCII to x'00'.
sgeransi.tab	Siemens German to ANSI. Control codes are converted from EBCDIC to x'01'. Control codes are converted from ASCII to x'00'.
danish.tab	Host code page 037 to pc code page 850. Control codes are converted.
swedish.tab	Host code page 037 to pc code page 850. Control codes are converted.

Enabling DBCS Processing

Use the International tab to enable Double Byte Character Set (DBCS) processing for converting data transferred between the CA-IDMS system and CA-IDMS Server. Click the **DBCS** radio button to enable the DBCS options, then select the type of DBCS used on the host. Click the **Default** button to use the default conversion tables.



Conversion Type one of the types installed on your system from the pull down list.

Conversion Table Path specifies the path of the subdirectory containing the DBCS conversion tables. This should normally be left as the default value.

Enable Half Width Katakana check this box to enable half width Katakana support when DBCS is enabled. All lowercase characters in CHAR and VARCHAR data are treated as half width Katakana. This does not affect GRAPHIC, VARGRAPHIC, and mixed data within SO and SI in CHAR and VARCHAR types. Only uppercase Roman text can be transferred between the mainframe and the PC when this option is enabled.

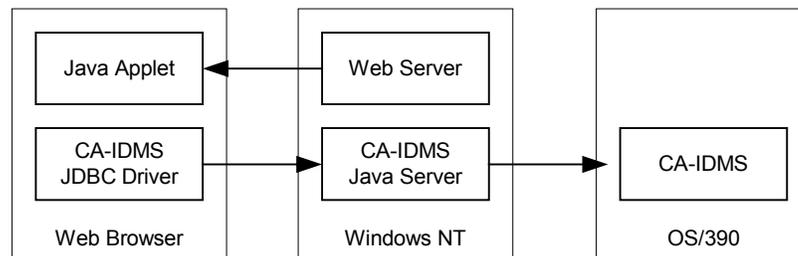
Configuring the Java Server

Use the Java Server tab to configure the Java Server. See Chapter 6, “Using the IDMS Java Server” for more information.

Using the CA-IDMS Java Server

Overview

The Java security model generally prevents an applet from opening a socket connection with a machine other than the web server from which it was loaded. The CA-IDMS Java Server is a Java application that acts like a "proxy" server for the CA-IDMS JDBC driver, allowing a JDBC applet running in a web browser to access a CA-IDMS System, as shown in the following diagram.



The Java Server is implemented as a Windows NT service and supports web servers running on Windows NT. A command line version is also provided that supports web servers running on other Java 1.1 platforms.

Installing the Java Server

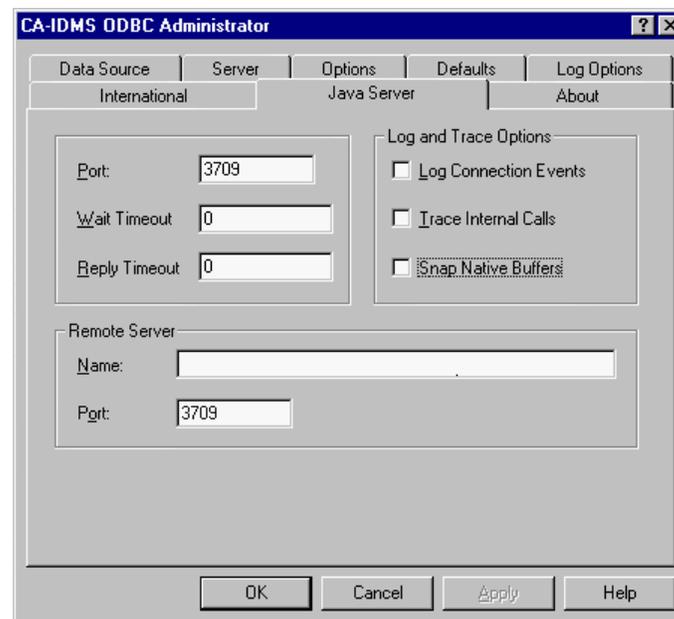
The CA-IDMS Java Server is installed automatically when the "Typical" option is selected during the installation of CA-IDMS Server, and is installed by default when "Custom" is selected. CA-IDMS Server can be reinstalled to add the Java Server, if needed. The CA-IDMS Java Server must be installed on the same machine as the web server.

Configuring the Web Server

The CA-IDMS JDBC driver classes must be installed in a directory accessible to web pages accessed from the web server. These classes are distributed as both standard Java archive (JAR) and as Microsoft cabinet files. These files, `idmsjabc.jar` and `idmsjabc.cab`, are installed into the `Java\classes` subdirectory of the CA-IDMS Server directory. You can define a virtual directory for the web server that points to this directory or copy the class files to an existing accessible directory.

Configuring the Java Server on Windows NT

The CA-IDMS ODBC Administrator is used to configure the Java Server service. Note that neither the CA-IDMS JDBC driver nor the Java Server actually use ODBC at runtime. To configure the Java Server, select any CA-IDMS data source from the ODBC Administrator, and then select the Java Server tab.



Port specifies the IP port that the Java Server will use to "listen" for connection requests. JDBC applications should specify this value in the URL. The default is 3709.

Wait Timeout is the number of seconds that the Java Server will wait for a request from the JDBC client. When this value is exceeded, the Java Server will consider the connection to have failed. Setting this to 0, the default, will cause the Java Server to wait indefinitely.

Reply Timeout specifies the number of seconds that the Java Server will wait for a response from the CA-IDMS server. When this value is exceeded, the Java Server will consider the connection to have failed. Setting this to 0, the default, will cause the Java Server to wait indefinitely.

Log Connection Events enables logging of connection requests and terminations by the Java Server to the Windows NT Application Event Log. By default only startup, shutdown, and error events are logged.

Trace Internal Calls enables tracing of debugging information to the CA-IDMS Server log file. Only internal method calls made by the Java code are traced. Use the Log Options to enable tracing of native method calls.

Snap Native Buffers enables display of the data buffers sent and received by the Java Server in the CA-IDMS Server log file.

It is possible to route JDBC connections to another Java Server before actually communicating with CA-IDMS. This can be useful when security requirements prevent the machine on which the web server is running from directly connecting to the mainframe. Such a "remote" server can be used by specifying:

Name is the DNS name or IP address of the remote Java Server machine.

Port is the listener port of the remote Java Server. The default is 3709.

Using the Java Server on Windows NT

Use the Windows NT Control Panel Services applet to start the CA-IDMS Java Server service. By default it is installed as a manually started Windows NT service. You can use the Control Panel to set it to start automatically.

The URL used by the applet identifies the address of the Java Server. An ODBC data source included in URL must be a "system" data source to be recognized by the Java Server. See Appendix B, JDBC Programmer Reference, for a description of the URL recognized by the CA-IDMS JDBC driver.

Using the Java Server on Other Platforms

To support web servers running on platforms other than Windows NT, the Java Server is also implemented as a command line application. This version is intended for application development and prototyping. It connects to CA-IDMS via a Java Server running on Windows NT, which is treated as a "remote server" as described above.

Future versions of CA-IDMS Server will implement Java Servers optimized for specific platforms other than Windows NT.

The Java Server application is provided as a JAR file in the CA-IDMS Server\Java\classes directory. This file, idmsjsrv.jar, can be copied to a web server machine that supports Java 1.1 and act as a CA-IDMS JDBC proxy server on that machine. Use the following command to start the Java Server using the Sun Microsystems JRE 1.1:

```
jre -cp <idmsclasspath>/idmsjsrv.jar ProxyMain <options>
```

where:

<idmsclasspath> identifies the location of the jar file
<options> controls the Java server.

Other Java implementations would use an equivalent command. The options are equivalent to those specified using the ODBC Administrator for the Windows NT service, and are specified by:

Option	Description
-?	prints usage information
-p <port>	specifies the listener IP port
-h <host>	specifies the remote host name
-r <port>	specifies the remote IP port
-w <seconds>	client wait timeout interval
-t <seconds>	server reply timeout interval
-v	enables "verbose" log output

Debugging options are:

```
-d [trace] [snap] [native] [snap] [buffer] [object] [remote] [util]  
-i <class> [<class>]...  
-x <class> [<class>]...
```

where:

d enables debugging with options:

- **trace** enables debug tracing
- **native** enables native trace
- **snap** enables object display
- **buffer** enables native buffer display
- **object** enables native object display
- **remote** enables remote on LocalHost (for testing)
- **util** sends trace to util log (only on NT)

i include <class> in trace

x exclude <class> from trace

Connecting to a Data Source Using ODBC

Overview

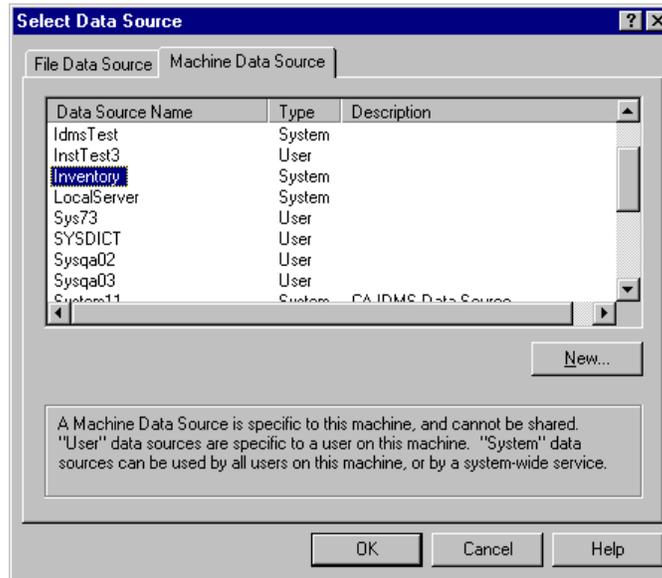
Many ODBC applications use the CA-IDMS DriverConnect dialog boxes to connect to a data source. If your application uses them, the CA-IDMS DriverConnect dialog boxes allow you to:

- Connect to an existing data source
- Add and connect to a new data source
- Connect dynamically to a data source that has not been defined previously

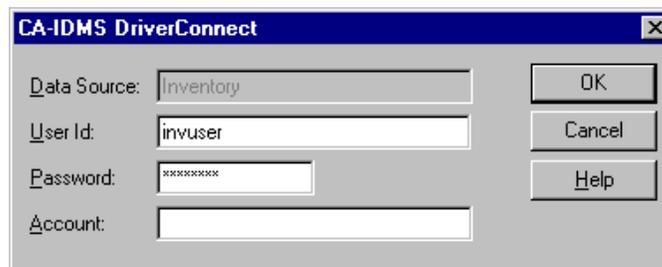
This chapter discusses the elements of data source connection and how to use the CA-IDMS DriverConnect dialog boxes to connect to a data source. These dialog boxes are implemented in the CA-IDMS ODBC driver. Although the JDBC driver uses the same types of information, it does not display any dialog boxes, leaving it to the JDBC application. See Appendix B for more information about connecting to a data source using JDBC.

Connecting to a Predefined Data Source

Many applications use the SQL Data Sources dialog box, shown below, to connect to a data source that has been defined with the ODBC Administrator tool.



The SQL Data Sources dialog box allows you to select the desired data source from a list of defined sources. Select the data source and then OK. The CA-IDMS DriverConnect dialog box appears.



In the CA-IDMS DriverConnect dialog box, specify the user ID, password, and account, if your application uses it. See Help for detailed information. Click OK to connect to the selected data source.

Connecting Dynamically to a Data Source Not Previously Defined

Some applications allow you to connect to a data source dynamically without defining or adding the data source first. The CA-IDMS DriverConnect dialog box appears if you can connect dynamically.

The screenshot shows the CA-IDMS DriverConnect dialog box with the following fields and values:

Section	Field	Value
Required	Dictionary	INVDICT
	Node Name	SYST0081
	User Id	invuser
	Password	*****
Optional	Task Code	CASERVER
	Account	
CCI Options	Wait Timeout	0
	Server Name	CCISERVER
	Server Port	1202

Supply the data source connection information that will be in effect for the duration of the session. This information is a subset of the data source definition information specified with the ODBC Administrator tool. See Help for detailed information about completing this dialog box.

Note: Unless you define the specified data source with the ODBC Administrator tool, the data source no longer exists when you end the session.

ODBC Programmer Reference

Overview

The Open Data Base Connectivity (ODBC) interface allows a Windows application to access different databases using SQL without specifically targeting any particular database. A module called an ODBC driver is used to link an application to a specific database.

The ODBC interface was developed by Microsoft and is aligned closely with the international-standard ISO Call-Level Interface.

Debugging User Sessions

CA-IDMS Server writes messages to the default PC log file, CAIDMS.LOG, specified in the Log and Trace Options dialog box described in Logging Errors and Trace Information of Chapter 5, "Managing Data Sources." These messages relay the status of the PC-to-mainframe database connection. Common messages relate to a user's authorization to sign on to the database, CCI timeouts, and unsuccessful connections because the CA-IDMS system is down.

Error Messages

Error messages returned by CA-IDMS Server have one of the following formats, depending on the component in which the error is detected:

```
[CA][IDMS ODBC Driver]Message text...  
[CA][IDMS ODBC Driver][CA-IDMS]Message text...
```

The first type of message is generated by the CA-IDMS ODBC driver as a result of an error condition it detected. The second type of message is generated as a result of an error detected within the ODBC data source, which includes CAICCI, CA-IDMS, and the network components.

Pseudoconversational Processing

When accessing the mainframe through the ODBC interface, CA-IDMS Server will automatically cause the backend task to pseudoconverse. This occurs whenever the ODBC driver would normally issue an autocommit (even if the application had turned autocommit processing off). This will also occur after each fetch and when a cursor is closed.

An ODBC application can cause the backend task to pseudoconverse at other times by executing a `SUSPEND SESSION` request. The session will be resumed automatically when the next ODBC request requiring mainframe access is issued.

Note: `SUSPEND SESSION` is not defined in the ODBC SQL grammar. Use of this command may restrict the portability of the application.

ODBC Conformance Levels

CA-IDMS Server 4.2 conforms to the ODBC 2.5 standard. Unless otherwise noted, all descriptions of ODBC in this document refer to ODBC 2.5.

Microsoft ODBC documentation specifies ODBC conformance in two areas: ODBC API conformance and ODBC SQL conformance. A driver must support all functionality in a conformance level in order to claim conformance to that level, but is not restricted from supporting a subset of the functionality of higher levels. ODBC defines functions that allow an application to determine the functionality supported by a driver in detail, including the API and SQL conformance levels, as well as specific API function, data type, and scalar function support.

API Conformance Levels

The ODBC API includes three conformance levels:

- Core API supports a set of core functions in the X/Open and SAG CLI specification.
- Level 1 supports core functionality plus an extended set of functions.
- Level 2 supports core and Level 1 functionality plus an extended set of functions.

The three conformance levels are described below.

Core API

The Core API corresponds to the functions in the X/Open and SQL Access Group (SAG) CLI specification, with a few minor differences. The Core API provides the minimum services to support dynamic SQL, including connection establishment and termination, SQL statement execution, retrieval of results, and transaction control.

The CA-IDMS ODBC driver supports all Core API functions. The Core API functions are as follows:

Core Function	Comments
ConfigDSN	For use by configuration programs.
SQLAllocConnect	
SQLAllocEnv	
SQLAllocStmt	
SQLBindCol	
SQLCancel	
SQLColAttributes	
SQLConnect	
SQLDescribeCol	
SQLDisconnect	
SQLError	
SQLExecute	
SQLExecDirect	
SQLFetch	
SQLFreeConnect	
SQLFreeEnv	
SQLFreeStmt	
SQLGetCursorName	Used by ODBC Cursor Library
SQLNumResultCols	
SQLPrepare	
SQLRowCount	
SQLSetCursorName	

Core Function	Comments
SQLSetParam	Supported by ODBC Driver Manager for ODBC 1.0 applications only. Replaced by SQLBindParameter (see below).
SQLTransact	

Level 1 API

The Level 1 extension to the API supports the Core functionality and also allows the application to set connection and statement options, retrieve information about driver and data source capabilities, and retrieve catalog information.

The CA-IDMS ODBC driver supports all Level 1 API functions. The Level 1 API functions are as follows:

Level 1 Function	Comments
SQLBindParameter	Replaces SQLSetParam
SQLColumns	
SQLDriverConnect	
SQLGetConnectOption	
SQLGetFunctions	Implemented by ODBC driver manager
SQLGetInfo	
SQLGetStmtOption	
SQLGetTypeInfo	
SQLParamData	
SQLPutData	
SQLSetConnectOption	
SQLSetStmtOption	No asynchronous operation; can set timeout, but it is ignored
SQLSpecialColumns	CA-IDMS does not support auto update columns
SQLStatistics	
SQLGetData	
SQLTables	

Level 2 API

The Level 2 extension to the API supports the Core and Level 1 functionality and also allows scrollable bulk cursors, retrieval of input parameter descriptions, and retrieval of additional catalog information.

The CA-IDMS ODBC driver supports the following Level 2 API functions:

Level 2 Function	Comments
SQLParamOptions	
SQLMoreResults	
SQLNumParams	

SQL Conformance Levels

The ODBC defines three conformance levels for SQL grammar:

- Minimum grammar supports minimal DDL, simple SELECT, INSERT, searched UPDATE and DELETE, simple expressions, and the CHAR data type.
- Core grammar supports the SQL statements and functionality proposed by the X/Open and SQL Access Group CAE specification (1992).
- Extended grammar supports common SQL extension supported by different database management systems.

The three conformance levels are described next.

Minimum SQL Grammar

The Minimum SQL Grammar is defined by ODBC to provide a basic level of conformance. It supports minimal DDL, simple SELECT, INSERT, searched UPDATE and DELETE, simple expressions, and the CHAR and VARCHAR data types.

CA-IDMS supports the Minimum SQL Grammar.

Integrity Enhancement Facility (IEF)

CA-IDMS Server supports referential integrity. Although CA-IDMS Server supports Integrity Enhancement Facility (IEF) functionality, it does not support the syntax defined in the ODBC specification, as noted in the following table. Implementation of IEF is optional for the ODBC driver and does not affect the SQL Conformance Level supported by the driver.

SQL Statement	Comments
CREATE TABLE clauses:	
DEFAULT	CA-IDMS supports WITH DEFAULT and allows default values of NULL, 0, or blank.
UNIQUE	CA-IDMS does not support specification of uniqueness constraints at either the column or table level. A unique index can be defined to provide the same effect.
PRIMARY KEY	CA-IDMS does not support specification of a primary key at either the column or table level. A unique index can be defined to provide the same effect.
REFERENCES	CA-IDMS does not support specification of referential constraints on the CREATE TABLE statement, at either the column or table level. The CREATE CONSTRAINT statement can be used to define referential constraints.
CHECK	CA-IDMS does not support specification of CHECK constraints at the column level. CHECK constraints can be specified at the table level.
DROP TABLE RESTRICT	CA-IDMS does not support the RESTRICT keyword, but does support CASCADE. The absence of CASCADE implies RESTRICT.
REVOKE CASCADE/RESTRICT	CA-IDMS does not support the CASCADE and RESTRICT options on REVOKE.

Core SQL Grammar

The Core SQL Grammar corresponds to the X/Open and SQL Access Group CAE draft specification (1991). Although similar to the ANSI SQL2 standard, it is not exactly the same. The Core SQL Grammar supports the Minimum SQL Grammar and adds more DDL, full SELECT, positioned UPDATE and DELETE, subquery, set expressions, and additional data types.

CA-IDMS supports the complete Core SQL Grammar.

The following ODBC SQL statements are converted by the CA-IDMS driver to CA-IDMS SQL:

SQL Statement	Comments
CREATE INDEX <i>user.index</i> DROP INDEX <i>user.index</i>	The CA-IDMS ODBC driver removes the qualifier from the index name if present.
GRANT UPDATE (<i>column-list</i>) REFERENCES (<i>column-list</i>)	CA-IDMS does not support column level security. The CA-IDMS driver removes the column list and grants UPDATE to all columns of the table.

Extended SQL Grammar

The Extended SQL Grammar supports the Minimum and Core SQL Grammar and adds outer joins, scalar functions, date/time literals, batch SQL statements, procedure calls, and additional data types.

CA-IDMS does not support the Extended SQL Grammar. Although CA-IDMS does support outer joins, it does not support the outer join syntax. CA-IDMS does not support procedure calls.

The ODBC driver supplied with CA-IDMS Server does not support outer joins and procedure calls. Scalar functions in escape sequences are passed to the data source unchanged except that the escape sequence itself is converted to spaces. SQL statements submitted in batch jobs are also not supported.

CA-IDMS supports data types that map to all ODBC data types.

The following statements specified in the ODBC Extended SQL syntax are not supported by the ODBC driver supplied with CA-IDMS Server Release 4.0:

Statement	Comments
ODBC procedure extension	CA-IDMS does not support procedures.
<i>outer join</i> LEFT OUTER JOIN	CA-IDMS supports outer join using the PRESERVE keyword, but the semantics are not identical.
<i>statement-list</i>	CA-IDMS and the CA-IDMS ODBC driver do not support this statement.
ODBC <i>scalar function extension</i>	CA-IDMS supports scalar functions that map to a subset of the ODBC scalar functions, but does not support the ODBC escape sequences for translation of scalar functions.
DELETE WHERE CURRENT OF <i>cursor</i>	Supported by using the ODBC Cursor Library.

Statement	Comments
SELECT FOR UPDATE	Supported by using the ODBC Cursor Library.
UPDATE WHERE CURRENT OF <i>cursor</i>	Supported by using the ODBC Cursor Library.

Database Type Mapping Between ODBC and CA-IDMS

The tables below describe how ODBC data types map to CA-IDMS database data types. The tables organize the data types by SQL conformance level. You can also use the SQLGetTypeInfo ODBC function to return detailed information about the mapping of ODBC and CA-IDMS data types.

Data Types

CA-IDMS to ODBC
Data Type Mapping

The following chart shows how CA-IDMS data types map to ODBC data types.

CA-IDMS Data Type	ODBC Data Type	Comments
BINARY	SQL_BINARY	
CHAR	SQL_CHAR	
CHARACTER VARYING	SQL_VARCHAR	Synonym for VARCHAR
DATE	SQL_DATE	
DECIMAL	SQL_DECIMAL	
DOUBLE PRECISION ¹	SQL_DOUBLE	
FLOAT ¹	SQL_FLOAT	
GRAPHIC ²	SQL_BINARY	DBCS disabled
	CAID_GRAPHIC	DBCS enabled
INTEGER	SQL_INTEGER	
LONGINT	SQL_BIGINT	
NUMERIC	SQL_NUMERIC	
REAL ¹	SQL_REAL	
SMALLINT	SQL_SMALLINT	

CA-IDMS Data Type	ODBC Data Type	Comments
TIME	SQL_TIME	
TIMESTAMP	SQL_TIMESTAMP	
UNSIGNED DECIMAL	SQL_DECIMAL	
UNSIGNED NUMERIC	SQL_NUMERIC	
VARCHAR	SQL_VARCHAR	
VARGRAPHIC ²	SQL_VARBINARY	DBCS enabled
	CAID_VARGRAPHIC	DBCS disabled
¹ Floating point conversion subject to rounding errors due to format differences.		
² If DBCS processing is enabled, GRAPHIC is mapped to CAID_GRAPHIC; VARGRAPHIC is mapped to CAID_VARGRAPHIC. These are CA-IDMS driver-specific data types.		

ODBC to CA-IDMS Data Type Mapping

The following chart shows how ODBC data types map to CA-IDMS data types.

ODBC Data Type	CA-IDMS Data Type
CAID_GRAPHIC ¹	GRAPHIC
CAID_VARGRAPHIC ¹	VARGRAPHIC
SQL_BINARY	BINARY
SQL_LONGVARBINARY	BINARY
SQL_CHAR	CHAR
SQL_DATE	DATE
SQL_DECIMAL	DECIMAL
SQL_DOUBLE	DOUBLE PRECISION
SQL_FLOAT ²	DOUBLE PRECISION
SQL_REAL ²	REAL
SQL_INTEGER	INTEGER
SQL_BIGINT	LONGINT
SQL_NUMERIC	NUMERIC
SQL_BIT	SMALLINT

ODBC Data Type	CA-IDMS Data Type
SQL_SMALLINT	SMALLINT
SQL_TINYINT	SMALLINT
SQL_TIME	TIME
SQL_TIMESTAMP	TIMESTAMP
SQL_LONGVARCHAR	VARCHAR
SQL_VARCHAR	VARCHAR

¹ Mapped when DBCS processing is enabled

² Floating point conversion subject to rounding errors due to format differences.

Driver-Specific Data Types

When DBCS processing is enabled, the CA-IDMS GRAPHIC and VARGRAPHIC data types are mapped to driver-specific ODBC SQL data types, as allowed by the ODBC 2.5 specification. These types are defined as CAID_GRAPHIC and CAID_VARGRAPHIC in the CAIDOOPT.H header file which is installed in the CA-IDMS Server directory. These types will be returned by SQLColumns, SQLDescribeCol, and SQLColAttributes, and they should be used with SQLBindParameter to define input parameters for GRAPHIC and VARGRAPHIC columns.

Since most applications are not specifically designed to handle DBCS data as defined by CA_IDMS, these types are treated in the same manner as SQL_CHAR and SQL_VARCHAR. The default C type for both is SQL_C_CHAR, and the precision is specified in bytes. Note that on CA-IDMS the length is specified in DBCS characters, which is half the precision specified using the CA-IDMS ODBC driver.

When DBCS is not enabled, GRAPHIC and VARGRAPHIC are mapped to SQL_BINARY and SQL_VARBINARY, with a default C type of SQL_C_BINARY and precision equal to the length in bytes.

SQLDriverConnect Connection String Format

CA-IDMS supports additional keywords for the SQLDriverConnect connection string.

The connection string has one of the following forms:
 DSN=data_source_name[;attribute=value[;attribute=value]...]
 DRIVER={CA-IDMS}[;attribute=value[;attribute=value]...]

Supported Attribute
Keywords and
Attribute Values

The table below provides a summary of the connection string attribute keywords and attribute values supported on the `SQLDriverConnect` function. This table includes both the keywords defined as part of the Microsoft ODBC specification and those defined as extensions for CA-IDMS Server. These keywords correspond to the fields in the `DriverConnect` dialog boxes as well as to the information used to define data sources and servers in the ODBC Administrator tool.

Keyword	Defined By	Attribute Value
DSN	Microsoft	Data source name
DRIVER	Microsoft	Driver name (cannot use with DSN)
DICT	CA	Dictionary name (use with DRIVER only)
NODE	CA	CA-IDMS System ID (use with DRIVER only)
TASK	CA	Alternate task code (use with DRIVER only)
UID	Microsoft	User ID
PWD	Microsoft	Password
ACCT	CA	Account information (if used)
CCINAME	CA	CCI host server name or IP address (optional, with driver only)
CCIPORT	CA	CCI host server port (optional, with driver only)
WAIT	CA	CCI reply wait timeout (optional, with driver only)

Example

An example of a connection string for CA-IDMS Server is shown below.

```
DSN=CA-IDMS database; UID=JELKA01; PWD=XYZZY; ACCT=R45-87
```

For More Information

Refer to the Microsoft *ODBC Programmer's Reference* for more information about calling the `SQLDriverConnect` function. See Chapter 6, "Connecting to a Data Source," earlier in this manual, and Help for information about the `DriverConnect` dialog boxes. See Chapter 5, "Managing Data Sources," for more information about attribute values.

Driver-Specific Connect Options

The ODBC options that can be specified for a data source using the ODBC Configurator can also be specified during program execution using `SQLSetConnectOption` and `SQLSetStmtOption`. These options and their values are defined in `CAIDOPT.H`, installed in the CA-IDMS Server directory.

Supported Isolation and Lock Levels

Transaction isolation is set with the SQLSetConnectOption ODBC API function. The default transaction isolation can be set using the ODBC Administrator (a CA-IDMS Server ODBC driver extension). The table below lists the transaction isolation levels that the CA-IDMS Server ODBC Driver supports and the corresponding CA-IDMS SET TRANSACTION options. The default is READ COMMITTED.

ODBC SQLConnect Option	CA-IDMS SET TRANSACTION Option
SQL_READ_UNCOMMITTED	TRANSIENT READ
SQL_READ_COMMITTED	CURSOR STABILITY

Bulk Insert Support

CA-IDMS Server supports the Core and Level 1 API functions listed in API Conformance Levels, earlier in this appendix. To facilitate Bulk Inserts, the CA-IDMS Server also supports these Level 2 functions:

- SQLParamOptions
- SQLMoreResults

To ensure that the ODBC driver takes advantage of the CA-IDMS INSERT...BULK feature, use parameter markers (?) in the Values clause of the Insert Into Tables statement; do not use a combination of parameter marks and constant values.

Retrieving Network Set Information

You can use the SQLExecuteDirect function with the following syntax to return information about network sets used to join network records accessed as SQL tables.

```
$SETS <owner> <table> <table>
```

where:

- <owner> is the name of the SQL schema that contains the names of the dictionary and network schema where the records are defined. This value applies to all tables and appears to the ODBC application as the TABLE_OWNER returned by SQLTables.

- *<table>* is the name of a record in the network schema. You can enter from zero to two *<table>* arguments. Each *<table>* argument must be unique and must be defined in the same network schema. This value appears to the ODBC application as the TABLE_NAME returned by SQLTables.

The *<owner>* and *<table>* name arguments are case-sensitive. The following table identifies what the result set contains depending on what you specify for the *<table>* arguments:

If You Specify	The Result Set Contains
No <i><table></i> argument	A list of all sets in the network schema referenced by <i><owner></i>
One <i><table></i> argument	A list of all sets in the network schema referenced by <i><owner></i> in which <i><table></i> is either the owner or a member
Two <i><table></i> arguments	A list of all sets in the network schema referenced by <i><owner></i> between the two <i><table></i> s, where either is the owner or member

The result columns are:

Column Name	Data Type	Description
SET_NAME	VARCHAR(18)	Network set name
SCHEMA_NAME	VARCHAR(18)	SQL schema name (ODBC owner)
OWNER_NAME	VARCHAR(18)	Network owner record name (ODBC table)
MEMBER_NAME	VARCHAR(18)	Network owner record name (ODBC table)

JDBC Programmer Reference

Overview

The JDBC interface allows a Java application to access different databases without specifically targeting any particular database. A set of classes called a JDBC driver is used to link an application to a specific database. The JDBC interface was developed by Sun Microsystems based on ODBC 2.5, and like ODBC, is consistent with the X/OPEN Call Level Interface (CLI).

This appendix provides information useful to developers of Java applications intended to access CA-IDMS databases. Additional information on the CA-IDMS implementation of JDBC can be found in the "javadoc" generated from the CA-IDMS JDBC driver source code. This HTML format documentation is installed, optionally, in the CA-IDMS Server directory and can be accessed from the Windows Start menu. A general familiarity with Java and JDBC is assumed.

JDBC Conformance

CA-IDMS Server 4.2 conforms to the JDBC 1.2 specification, which is included in the Java 1.1. Unless otherwise noted, all descriptions of JDBC in this document refer to JDBC 1.2.

API Conformance

JDBC does not define "conformance levels" in the same sense that ODBC does. A JDBC driver must implement all methods defined in the specification, however, the driver can throw an exception, or return some sort of 0 or null value to indicate that it cannot do what the method requires. The CA-IDMS JDBC driver implements the following methods only to satisfy the JDBC specification:

Class.method	Comment
CallableStatement.execute	CA-IDMS does not support stored procedures.
CallableStatement.executeQuery	
CallableStatement.executeUpdate	
CallableStatement.getBigDecimal	
CallableStatement.getBoolean	
CallableStatement.getByte	
CallableStatement.getBytes	
CallableStatement.getDate	
CallableStatement.getDouble	
CallableStatement.getFloat	
CallableStatement.getInt	
CallableStatement.getLong	
CallableStatement.getObject	
CallableStatement.getShort	
CallableStatement.getString	
CallableStatement.getTime	
CallableStatement.getTimestamp	
CallableStatement.registerOutParameter	
CallableStatement.wasNull	
Connection.setCatalog	Catalog name not used to qualify tables.
DatabaseMetaData.getCatalogs	
DatabaseMetaData.getColumnPrivileges	Information not available in catalog.
DatabaseMetaData.getCrossReference	
DatabaseMetaData.getExportedKeys	
DatabaseMetaData.getImportedKeys	
DatabaseMetaData.getPrimaryKeys	
DatabaseMetaData.getProcedures	
DatabaseMetaData.getProcedureColumns	

Class.method	Comment
DatabaseMetaData.getTablePrivileges	
DatabaseMetaData.getVersionColumns	
ResultSetMetaData.getCatalog	Information not available in SQLDA.
ResultSetMetaData.getSchemaName	
ResultSetMetaData.getTableName	
Statement.cancel	Client cannot cancel mainframe task.
Statement.getMoreResults	Batched statements not supported.

Refer to the "javadoc" for more detailed information, including the specific values returned and exceptions thrown by the CA-IDMS JDBC Driver methods.

SQL Conformance

To be "JDBC COMPLIANT" a JDBC driver must support ANSI SQL-92 Entry Level. This is consistent with ODBC 3.0. With a few minor exceptions, CA-IDMS conforms to the ANSI SQL-92 entry level standard. Both the the CA-IDMS ODBC and JDBC drivers pass most SQL statements to CA-IDMS essentially unchanged other than converting escape sequences into CA-IDMS equivalents. For more information about SQL conformance refer to Appendix A.

Database Type Mapping Between JDBC and CA-IDMS

The tables below describe how JDBC data types map to CA-IDMS database data types. Java applications can use the DatabaseMetaData getTypeInfo method to return detailed information about the mapping of JDBC and CA-IDMS data types.

Data Types

CA-IDMS to JDBC
Data Type Mapping

The following chart shows how CA-IDMS types map to JDBC data types when data is returned in a result set.

CA-IDMS Data Type	JDBC Data Type	Comments
SMALLINT	SMALLINT	
INTEGER	INTEGER	
LONGINT	BIGINT	
REAL	REAL	
FLOAT	REAL	Precision < 25.
FLOAT	FLOAT	Precision > 24.
DOUBLE PRECISION	DOUBLE	
DECIMAL	DECIMAL	
UNSIGNED DECIMAL	DECIMAL	
NUMERIC	NUMERIC	
UNSIGNED NUMERIC	NUMERIC	
CHAR	CHAR	
GRAPHIC	CHAR	DBCS must be enabled.
VARCHAR	VARCHAR	
VARGRAPHIC	VARCHAR	DBCS must be enabled.
BINARY	BINARY	
DATE	DATE	
TIME	TIME	
TIMESTAMP	TIMESTAMP	

JDBC to CA-IDMS
Data Type Mapping

The following chart shows how JDBC data types map to CA-IDMS types when a parameter value is set.

JDBC Data Type	CA-IDMS Data Type
BIT	SMALLINT
TINYINT	SMALLINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	LONGINT
REAL	REAL
FLOAT	DOUBLE PRECISION
DOUBLE	DOUBLE PRECISION
DECIMAL	DECIMAL
NUMERIC	NUMERIC
CHAR	CHAR
VARCHAR	VARCHAR
LONGVARCHAR	VARCHAR
BINARY	BINARY
VARBINARY	BINARY
LONGVARBINARY	BINARY
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP

Connection Parameters

This section describes the information needed to connect to a CA-IDMS database using JDBC, including the URL formats and DriverProperties recognized by CA-IDMS JDBC driver. This information is passed to the JDBC Driver Manager get Connection method.

IDMS URL Format

A URL (Uniform Resource Locator) is used to locate a resource on the Internet. A URL always begins with a protocol followed by a colon, such as "http:" or "ftp:", the rest of the string is defined by the protocol. In keeping with the Internet orientation of Java and JDBC, a URL is used to identify a database. The JDBC specification defines conventions for the format of a JDBC URL. Each JDBC driver defines the actual format of the URLs that it recognizes. The general format of a JDBC URL is:

<protocol>: <subprotocol>: <subname>

<protocol> is always "jdbc". <subprotocol> and <subname> are defined by the JDBC driver.

The CA-IDMS JDBC driver recognizes two URLs with <subprotocol> "idms". The location of the native CA-IDMS SQL client interface and the data source or dictionary name specified by the <subname>.

```
jdbc:idms:<database>  
jdbc:idms://<hostname>:<port>/<database>
```

The first format is used when the JDBC driver is running on the same machine as the native CA-IDMS SQL client interface. This is typically the case when a Java application is running on the machine where CA-IDMS Server is installed. The JDBC driver calls the native interface directly.

The second format is used when the JDBC driver is running on a different machine than the native CA-IDMS SQL client interface. This is typically the case when a Java applet is running in a web browser and CA-IDMS Server is installed on the machine where the web server is running. The JDBC driver communicates with the CA-IDMS Java Server, which calls the native interface directly. <hostname> is the DNS name or IP address of the web server, and <port> is the IP port that was specified as the CA-IDMS Java Server listener.

In either case <database> can be an ODBC data source name or the dictionary name of the catalog containing the table definitions. When <database> is an ODBC data source name, the actual dictionary and physical connection information are resolved by the CA-IDMS Server native interface, and must be defined on the system where the native code runs. When <database> is a dictionary name the physical connection information is specified by DriverPropertyInfo objects.

DriverPropertyInfo

JDBC DriverPropertyInfo objects are analogous to the connection attributes used by the ODBC SQLDriverConnect and SQLBrowseConnect functions. For the CA-IDMS JDBC driver they are used to specify the user id, password, and optional accounting information. They can also be used to specify physical connection information, allowing an application to connect to a CA-IDMS database without requiring an ODBC data source to be defined. CA-IDMS supports the following driver properties:

Name	Description
user	The user id used to sign on to CA-IDMS. Always required.
password	The password associated with the user id. Required to connect to a secured CA-IDMS system.
account	Accounting information. This is an optional feature that may be used by the CA-IDMS system. A user exit must be installed on the DC system to process the information. See the Appendix D for more information.
node	The DC NODE name, it identifies the CA-IDMS system that contains the data base. This allows a connection to be established without defining an ODBC data source. Use of this property implies that the subname contains a DICTIONARY name.
via	The NODE name of an intermediate CA-IDMS DC system used to route requests to the target system. This is used when a physical connection cannot be established directly to the CA-IDMS DC system containing the SQL database. Only valid if node is specified.

Name	Description
task	The DC TASK code that invokes the internal CA-IDMS SQL server interfaces. Only valid if node is specified.
host	The DNS name or IP address of the CCI host server for use by the native CA-IDMS Server SQL client interface. Only valid if node is specified. Usually the default is used.
port	The IP port of the CCI host server. Only valid if host is specified. Usually the default is used.

Dynamic Positioned Updates

The CA-IDMS JDBC driver supports positioned updates and deletes in dynamic SQL when connected to a CA-IDMS Release 14.0, or later, system. For prior releases the ResultSet setCursorName and getCursorName methods are implemented only to conform to the JDBC specification, and are not used internally.

In order to use positioned updated and deletes the FOR UPDATE clause must be specified in the SQL query statement. According to the JDBC specification the SQL query must have the form:

```
SELECT FOR UPDATE ... FROM ... WHERE ...
```

CA-IDMS expects the FOR UPDATE clause at the end of the syntax:

```
SELECT ... FROM ... WHERE ... FOR UPDATE
```

The CA-IDMS JDBC driver will recognize the clause in either location, and move it to the end before passing the syntax to CA-IDMS, if necessary.

To optimize performance, the CA-IDMS JDBC driver normally attempts to fetch up to 100 rows at a time. Since row currency would be at the last row, issuing a positioned update or delete would not have the expected effect, so specifying the FOR UPDATE clause causes the driver to fetch one row at a time.

The JCF Demo

The CA-IDMS Java Command Facility (JCF) is provided as an example using the “Swing” classes and JDBC to access CA-IDMS databases. JCF can be used as an applet in a browser or as a standalone Java application. Included are the class files, source code, and sample HTML and batch files.

Registry Information

Overview

The registry is a database used by Windows 95 and Windows NT to store system and application information. It replaces the "ini" files used in Windows 3.1.

Registry Information

This section describes the information stored in the registry and used by CA-IDMS Server 4.2. This information is provided to help you identify problems that may arise with CA-IDMS Server 4.2. The registry information is maintained using the 32-bit ODBC Administrator, which is available from the Control Panel. Unlike ini files, it cannot be edited directly, but it can be edited using the registry editor provided by Microsoft. Editing the registry directly is not recommended, since an error can disable not only CA-IDMS Server, but also Windows itself.

The registry is structured as a hierarchical database, with keys, subkeys, and values. At the top level are four or five keys, two of which are used by ODBC and CA-IDMS Server. HKEY_LOCAL_MACHINE contains information about hardware and software common to all users of the machine. HKEY_CURRENT_USER contains preferences and application settings for the current user. A subkey is analogous to a directory path and is specified in a similar fashion. The following are the subkeys used by ODBC and CA-IDMS:

- HKEY_LOCAL_MACHINE\Software\ODBC\ODBCINST.INI
- HKEY_LOCAL_MACHINE\Software\ODBC\ODBC.INI
- HKEY_CURRENT_USER\Software\ODBC\ODBC.INI
- HKEY_LOCAL_MACHINE\Software\ComputerAssociates\CA-IDMS

Under each of these keys are subkeys corresponding to the section names used in ini files. At the lowest level are value names, corresponding to the key names used in ini files, and the values themselves. The remainder of this appendix describes the information in these subkeys.

HKEY_LOCAL_MACHINE\Software\ODBC\ODBCINST.INI

This section contains information about the ODBC drivers installed on the machine and corresponds to the ODBCINST.INI file used in Windows 3.1. The CA-IDMS Server 4.2 installer program adds the information for the CA-IDMS ODBC driver using the ODBC installer DLL when the product is installed. This key contains the following information (note that the Value Data column assumes that CA-IDMS is the only installed ODBC Driver). These values are documented in more detail in the *ODBC SDK Reference*.

Subkey	Value Name	Description
ODBC Core	UsageCount	Driver manager usage count.
ODBC Drivers	CA-IDMS	There is an entry for each installed ODBC driver, where the value name is the driver name and the value data is "Installed".
CA-IDMS		There is a subkey for each installed driver, where the name of the subkey is the name of the driver.
	APILevel	Driver ODBC API conformance level. For CA-IDMS, the value is "1".
	ConnectFunctions	Connect functions supported by driver. For CA-IDMS, the value is "YYN".
	Driver	Driver DLL name and path.
	DriverODBCVer	Version of ODBC supported by driver. For CA-IDMS, the value is "02.50".
	FileExtns	Not used for CA-IDMS.
	FileUsage	Not used for CA-IDMS.
	Setup	Driver setup (configurator) DLL name and path.
	SQLLevel	Driver SQL conformance level. For CA-IDMS, the value is "1".
	UsageCount	Driver usage count.
Default	Driver	Name of ODBC driver for default data source.

HKEY_LOCAL_MACHINE\Software\ODBC\ODBC.INI

This section contains information about system data sources, which are available to all users of the system, as well as system services, and corresponds to the ODBC.INI file used in Windows 3.1. The CA-IDMS Server 4.2 configurator maintains this information using the ODBC installer DLL when the ODBC administrator is used from the Control Panel. The ODBC .INI key contains the following subkeys and values:

Subkey	Value Name	Description
ODBC Data Sources	<DSN>	There is an entry for each data source where the Value Name is the data source name and the Value Data is the driver name; for CA-IDMS, this is "CA-IDMS".
<DSN>		There is a subkey for each data source where the subkey name is the data source name.
	Driver	Driver DDL name and path, copied from the ODBCINST.INI key.
	Dictionary	Identifies the DBNAME or segment name of the CA-IDMS dictionary as defined in the CA-IDMS DBNAME table on the target CA-IDMS system. This value comes from the Dictionary text box in the CA-IDMS ODBC Administrator dialog box. If no dictionary name is specified, the default is the first eight characters of the data source name.
	Server	Specifies the user-defined server name.
Default		The Default data source can contain the same values as other data source definitions.

HKEY_CURRENT_USER\Software\ODBC\ODBC.INI

This section contains information about user data sources, which are available only to the currently signed-on users of the system and corresponds to the ODBC.INI file used in Windows 3.1.

The CA-IDMS Server 4.2 configurator maintains this information using the ODBC installer DLL. The structure of the information under this key is the same as the ODBC.INI subkey of HKEY_LOCAL_MACHINE.

HKEY_LOCAL_MACHINE\Software\ComputerAssociates\CA-IDMS

This section contains information about global options, data source default options, and server definitions for all users and system services, and corresponds to the CAIDDSI.INI file used in Windows 3.1.

Data source information that was maintained in the CAIDDSI.INI file is now stored under the ODBC.INI keys, in order to support user and system data sources. The CA-IDMS Server 4.2 configurator maintains this information directly.

The CA-IDMS key contains information that CA-IDMS uses to establish a connection to a CA-IDMS database anywhere in a network of CA-IDMS systems.

The CA-IDMS key can contain these subkeys:

- **Servers** subkey to associate a server name with an ODBC driver name
- **Server** *server_name* subkey for each server to define its database access path information
- **Options** subkey for log settings, language settings, and path
- **Defaults** subkey for default database settings

The Servers, Server Name, and Options subkeys are described next.

Servers

The Servers Subkey lists all servers defined using the CA-IDMS ODBC Administrator dialog box under the Microsoft ODBC Administrator application.

The Servers Subkey contains the following parameters:

Value Name	Value
<i>server_name</i>	CA-IDMS

Values

Server_name is always CA-IDMS, the name of the ODBC driver.

Server *Server_name*

The Server *Server_name* subkey contains information describing a CA-IDMS system.

The Server *Server_name* subkey contains the following parameters:

Value Name	Value
AlternateTask	<i>task_code</i>
Node	<i>via_node_name</i>
Resource	<i>node_name</i>
CCI Server Name	<i>cci_name</i>
CCI Server Port	<i>port</i>
Wait Timeout	<i>wait</i>

Values

Server_name is a server name listed in the Servers section and referenced by a datasource definition.

Node_name optionally identifies the value of SYSTEMID as specified in the SYSTEM statement of the system generation of the target system. If a node name is not specified, CA-IDMS Server uses the first eight characters of the *server_name* to identify the target system.

Task_code optionally identifies an alternate task that defines the resource limits and timeout values for a session. The default is CASERVER. If you accept the default or define another task, be sure it is defined as a task on the CA-IDMS System Generation TASK Statement. This value comes from the Task Code text box in the CA-IDMS Server Administration property sheet.

Via_node_name optionally specifies the node that CCI will establish a connection with. The system identified by via node must contain a RESOURCE table entry for the system identified by node name. Use this option when the system containing your tables does not directly communicate with CCI.

Cci_name identifies the DNS name or IP address of where the CCITCP Server is running. If this is not specified, the default server defined for CCI is used.

Cci_port optionally identifies the IP port that identifies the CCITCP Server on the node defined by *cci_name*. If not specified, the default port defined for CCI is used. This is normally 1202 and usually should not be specified here.

Wait specifies the number of seconds that CCI will wait for a response from the CA-IDMS system. When this interval is exceeded, CA-IDMS Server will consider the connection to have failed. Setting this to 0 will cause CCI to use the default value specified using the CCI Configurator.

For more information on resource limits for external user sessions, see *CA-IDMS System Generation* and *CA-IDMS System Operations*.

Options

The Options subkey contains global options. This key is updated with the values from the options you select in the Defaults, Log Options, and International tabs in the CA-IDMS ODBC Administrator Property Sheet dialog box.

The Options subkey contains the following parameters for global options:

Value Name	Value
AsciiEdcdicTables	<i>translation_table_name</i>
CmTrace	<i>integer_value</i>
DbcsPath	<i>dbcs_path</i>
DbcsType	<i>dbcs_type</i>
DnsTrace	<i>integer_value</i>
DtsTrace	<i>integer_value</i>
FdeTrace	<i>integer_value</i>
JdbcTrace	<i>integer_value</i>
LogFile	<i>log_file_name</i>
LogOptions	<i>log_option_values</i>
NdlTrace	<i>integer_value</i>
JdbcTrace	<i>integer_value</i>
OdbcTrace	<i>integer_value</i>
SQLTrace	<i>integer_value</i>
UtilTrace	<i>integer_value</i>
MultiThread	<i>thread_value</i>
FetchSuspend	<i>suspend_flag</i>

Values

log_file_name specifies the name of the log file, if you entered something other than the default log name. This value comes from the Log File Name text box in the Log Options tab accessed from the CA-IDMS ODBC Administrator property sheet.

log_option_values specifies log options with 1—Appends logging information to the log file each time an ODBC session is started.

translation_table_name specifies the name of the Country Extended Code Page translation table you selected to convert EBCDIC data on the server to ASCII data on the PC and vice versa. The value comes from the International tab accessed from the CA-IDMS ODBC Administrator property sheet.

integer_value specifies the flag bits used to control tracing. The Options section can also contain trace flags that Computer Associates Technical Support uses to resolve CA-IDMS Server problems. The *integer_value* must be in the range of 0, which signifies all options off, to 65535, which signifies all options on. This value can be specified as a decimal or hexadecimal integer. Descriptions of the individual bit flags are as follows:

JdbcTrace (ca.idms.*):	any non-zero value enables tracing
OdbcTrace (CAIDOD32.DLL):	0x0002 // Trace internal functions 0x0004 // Trace function parms 0x0008 // Trace thread locks 0x0010 // Snap SQL syntax 0x0100 // Snap environment block 0x0200 // Snap connection block 0x0400 // Snap statement block 0x0800 // Snap SQLDA
SqlTrace (CAIDQC32.DLL):	0x0002 // Time SQL calls 0x0004 // Snap SQL SQLSID0x0008 // Snap SQL DSICB 0x0010 // Snap SQL SQLCA 0x0020 // Snap SQL SQLCIB 0x0040 // Snap SQL SQLPIB 0x0080 // Snap SQL parm buffer 0x0100 // Snap SQL tuple buffer 0x0200 // Snap SQL input SQLDA 0x0400 // Snap SQL output SQLDA 0x0800 // Snap SQL syntax string 0x4000 // Trace server calls 0x8000 // Snap server interface blocks
DtsTrace (CAIDTD32.DLL):	0x0002 // trace external calls 0x0004 // trace events 0x0008 // trace events 0x0010 // snap user data arrays 0x0020 // trace events 0x0040 // snap PCE 0x0080 // snap LCE
DnsTrace (CAIDTD32.DLL):	0x0010 // snap unconverted send data 0x0020 // snap converted send data 0x0040 // snap received data 0x0080 // snap converted received

CmTrace (CAIDTD32.DLL):	0x0001 // trace CCI and internal function calls 0x0002 // elapsed CCI call timings
FdeTrace (CAIDFD32.DLL):	0x0001 // trace external generate calls (for precompiler) 0x0002 // trace external convert calls 0x0004 // trace external ASCII-EBCDIC conversion calls 0x0010 // trace internal calls 0x0100 // snap format descriptors 0x1000 // snap input (unconverted) data 0x2000 // snap output (converted) data
UtilTrace (CAIDUT32.DLL):	0x0001 // Trace external calls 0x0002 // Trace internal calls 0x0004 // Trace DllEntry calls

dbcs_path specifies the path to the DBCS translation tables. This is normally the direction specified when CA-IDMS Server 4.2 was installed.

dbcs_type specifies the integer value identifying the Double Byte Character Set Language, as defined by the DBCS Types subkey.

thread_value controls whether CA-IDMS Server will process ODBC connections or multiple threads concurrently. 1 enables multi-threaded access; 0 disables it.

suspend_flag is a DWORD value that determines whether the CA-IDMS ODBC driver automatically issues a SUSPEND SESSION command after every “bulk” FETCH. This “pseudo-conversational” processing reduces resource usage by the CA-IDMS system, but requires the exchange of an additional message between the PC and mainframe, which can impact performance when retrieving a very large result set. The automatic SUSPEND should normally be enabled (and is by default), but can be disabled by setting the value to 0. This setting is ignored when the data source is on a CA-IDMS Release 14.1 system, since the extra message is not needed.

The ODBC configurator is normally used to enable and disable tracing. Since tracing can add overhead and affect performance, it should be disabled under normal circumstances.

Defaults

The Defaults subkey contains the default ODBC options, used when adding a new data source. This key is updated from the options you select in the Defaults tab in the CA-IDMS ODBC Administrator property sheet.

The Defaults subkey contains the following parameters for default options for new data sources; Data Source subkeys in the ODBC.INI key also contain these parameters.

Value Name	Value
AccessibleTables	0/1
AccountPrompt	0/1
CacheSQLTables	0/1
CatalogTable	<i>view_name</i>
CommitBehavior	0/1/2
EnableEnsure	0/1
FetchDouble	0/1
FetchRows	<i>number_of_rows</i>
Path	<i>path_name</i>
ReadOnly	0/1
TxnIsolation	0/1

Values

AccessibleTables tells the ODBC driver whether to use the SYSCA.ACCESSIBLE_TABLES view (or another view defined by you) for the SQLTables function. The value 0 indicates do not use the view; the value 1 indicates use the view. This value comes from the Use Accessible Tables View Name check box.

AccountPrompt tells the ODBC driver to prompt for information if the ACCT keyword is not supplied in the connection string passed to SQLDRiverConnect. See Appendix C, "Passing Accounting Information to CA-IDMS," for more information.

CacheSQLTables instructs CA-IDMS Server to cache the table list returned from an SQLTables call. This value comes from the Cache SQL Tables check box. The value 1 indicates that the option is on; the value 0 indicates that the option is off.

CatalogTable=*view_name* specifies the name of the view to use for the SQLTables function if you specified something other than the default view name. This value comes from the Accessible Tables View Name text box.

CommitBehavior determines how a COMMIT operation affects cursors in CA-IDMS. The value 0 indicates Close and Delete Cursors, which closes and deletes cursors. The application must prepare and execute the next statement. The value 1 signifies Close Cursors, which closes cursors. Applications can execute a statement without preparing the statement again. The value 2 signifies Preserve Cursors, which preserves cursors in the same position as before the COMMIT operation. The application can execute or fetch without preparing the statement again. This value comes from the Commit Behavior list box.

EnableEnsure instructs the CA-IDMS Server to honor the ENSURE parameter of the SQLStatistics function call. This value comes from the Enable Ensure check box. The value 1 indicates that the option is on; the value 0 indicates that the option is off.

FetchDouble specifies whether four-byte floating point numbers should be converted to eight-byte floating point before CA-IDMS Server fetches the data. The value 1 indicates that the option is on; the value 0 indicates that the option is off. This value comes from the Fetch Real as Double check box in the Set Administrator Defaults dialog box accessed from the Options list in the CA-IDMS ODBC Administrator property sheet.

number_of_rows specifies the number of database rows CA-IDMS Server should fetch at a time. The default is 100. This value comes from the Bulk Fetch Row Count text box in the Set Administrator Defaults dialog box accessed from the Options list in the CA-IDMS ODBC Administrator property sheet.

path_name specifies the directory where files used by CA-IDMS Server are installed. The default is *x:* \CA_APPSW\CAID, where *x:* identifies the disk drive. Note that the ODBC and CA-IDMS Server DLLs are installed in the Windows system directory.

ReadOnly specifies whether the Windows application can both read and update the CA-IDMS database or only read it. The value 0 indicates that it can read and update (READ WRITE); the value 1 indicates that it can only read it (READ ONLY). This value comes from the Access Mode list box in the Set Administrator Defaults dialog box accessed from the Options list in the CA-IDMS ODBC Administrator property sheet.

TxnIsolation specifies the degree to which your transactions are impacted by (and impact) other users accessing the same data. The value 1 indicates READ UNCOMMITTED; the value 2 indicates READ COMMITTED. The default is READ COMMITTED. This value comes from the Transaction Isolation list box.

Specify these options in the Defaults tab in the CA-IDMS ODBC Administrator dialog box. They can also be specified for an individual data source using the Options tab. In this case, the values are stored under the data source name subkey in the registry.

DBCS Types The DBCS Types subkey identifies the languages that have DBCS support. The values are added when CA-IDMS Server is installed.

Proxy

The Proxy subkey contains information used to the configure the CA-IDMS Java Server, and contains the following parameters:

Value Name	Value
Port	<i>integer</i>
RemoteHost	<i>host_name</i>
RemotePort	<i>integer</i>
TimeOut	<i>integer</i>
WaitTimeOut	<i>integer</i>
Trace	0 1
Snap	0 1
Verbose	0 1

Values **Port** is the IP port that the Java Server listens on for connection requests. The default value is 3709.

RemoteHost is the DNS name or IP address of another Java Server, used to forward packets to the CA-IDMS system. This is optional.

RemotePort is the IP port address of the remote host. The default value, if used, is 3709.

TimeOut is the number of seconds that the Java Server will wait for a response from the CA-IDMS server. The default, 0, will cause the Java Server to wait indefinitely.

WaitTimeOut is the number of seconds that the Java Server will wait for a request from the JDBC client. The default, 0, will cause the Java Server to wait indefinitely.

Trace enables tracing of internal function calls. Output is written to the CA-IDMS log file.

Snap enables display of data buffers sent and received in the CA-IDMS log file.

Verbose enables logging of connection requests to the Windows NT application event log.

Passing Accounting Information to CA-IDMS

Overview

This appendix describes how to use CA-IDMS Server to pass accounting information from the PC client to the backend CA-IDMS system.

Supplying Accounting Information

You can supply as many as 32 bytes of accounting information by using one of the following methods:

- Enter the value in the Account field of the ODBC DriverConnect dialog box.
- Pass the value as a connection attribute with the key "ACCT" to the ODBC SQLDriverConnect function.
- Pass the value as a Driver Property Info object with the key "acct" to the JDBC DriverManager getConnection method.

The first character of accounting information must be a space; otherwise, it will be ignored.

How Accounting Information Is Used

The accounting information is passed to the backend and is stored in an area accessible from the PTE for use by accounting and security exits. For example, the information could be used by either exit 4 or 5 to change header information in the statistics block described by #STRDS.

The following table describes how the accounting data can be found.

Field Name	Control Block	Comments
TCELTEA	TCE	Points to the LTE. Determine whether it is a CCI-related LTE (type is LTEBULK).
LTEPTEA	LTE	Points to the PTE.
PTELACCT	PTE	Points to a 32-byte accounting information area. This area will be binary zeros if no accounting information was passed from the PC or if the frontend is not a PC.

Example

Sample code for locating accounting information is shown below.

```

USING TCE,R9
  L   R5,TCELTEA           Get the LTE
USING LTE,R5
  CLI LTETYP,LTEBULK      Is this a BULK type
  BNE RETURN              No...Return
  L   R6,LTEPTEA          Get the PTE address
USING PTE,R6
  L   R7,PTELACCT         R7 points to 32-byte area
    
```

Configuring CAICCI for TCP/IP

Overview

The CA90s version of CAICCI/PC now includes multi-threading support when configured for TCP/IP. This version can be downloaded from the latest CA90s tape. For convenience, the CCI subdirectory of the CA-IDMS Server 4.2 CD contains a self extracting file that can be used to install CAICCI/PC. The following information is included as a brief overview of how to configure and use CAICCI/PC with TCP/IP. For further information, refer to CA90s documentation.

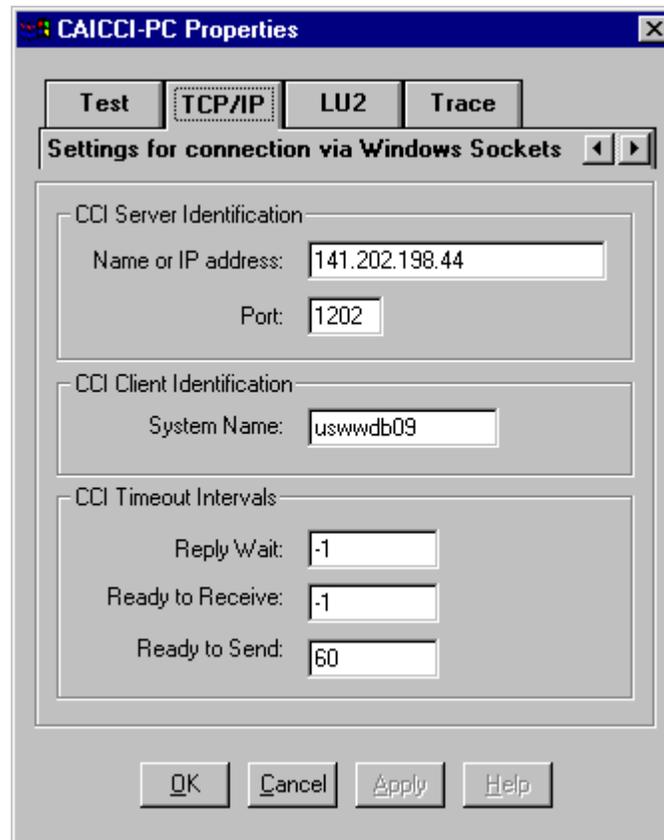
Installation Notes

It is not necessary to reinstall CAICCI/PC when upgrading from CA-IDMS Server 4.1 after installing the CA90s version of CAICCI/PC multi-threaded TCP/IP support may not be enabled. To correct this, use the CCI Configurator to modify any TCP/IP parameter and press OK or Apply.

CA-IDMS Server 4.1 installed the multi-threaded TCP/IP version of CAICCI/PC in the Windows System directory, where it was accessible to all Windows programs. The CA90s version is installed in the CA_APPSW directory. After installing the CA90s version delete the file caic3cci.dll from the Windows System (or System 32) directory. In order to use this version of CAICCI/PC with Windows NT Services, the CA_APPSW directory must be included in the PATH specified in the system environment variables. The CAICCI/PC installer may only update the PATH in the current user's environment variables.

Specifying Protocol Parameters for TCP/IP

You can run the CCI Configurator from the Start Menu.



Name or IP address In the CCI Server Identification box, enter the name or Internet address of the server which acts as a gateway for CCI requests.

Normally, you enter an Internet address for the CCI server. However, if you want to use a logical name, you must be running a TCP/IP transport that has been configured to work with a name server, and the name you enter is defined to the name server.

Port Enter the port number that the CCI server expects to use. The default is 1202.

When CAICCI-PC is configured to provide multi-threading support, the application can override the CCI server name and port.

System Name In the CCI Client Identification box, you can override the System Name defined by TCP/IP and enter a different name to identify your PC.

Note: By default, CCI uses the name that TCP/IP defines, truncated to eight characters. This truncation may create a problem of identical IDs. To avoid this, or to use a preferred ID, enter a new name in the **System Name** box.

After identifying the CCI Server, the Port number, and the CCI Client system name, the TCP/IP configuration is set. The CAICCI Configurator will set the remaining parameters based on the selected software. Click OK to exit this dialog box.

The following timeout options are used when CAICCI-PC is installed to provide multi-threading support.

Reply Wait Enter the number of seconds for CAICCI to wait for a response from the other CCI application. When this limit is exceeded the local application may continue to wait or assume an error and terminate the connection. Setting this value to -1 will cause CAICCI-PC to wait indefinitely. Specifying 0 will cause CAICCI-PC to poll for a response indefinitely and is supported for compatibility with previous versions of CAICCI-PC. Using a nonzero timeout value is more efficient and results in lower processor use by the application.

Ready to Receive Enter the time interval that CAICCI-PC will wait for the individual packets. A value of -1 will cause CAICCI to wait indefinitely, and 0 will cause it to poll indefinitely. Setting -1 is recommended in most situations.

Ready to Send Enter the time interval that CAICCI-PC will wait to be able to send data over a connection. Values are entered as they are for ready to receive. The default is 60.

Tracing a Communications Problem

CAICCI allows you to optionally generate an internal trace file that can be used by product support in the event that a communication problem occurs. To enable the creation of this file, select Trace in the CAICCI-PC Properties dialog box.



The default filename for the trace file is CCITRACE.TRC. This file is already in ASCII format and does not have to be converted.

To change the name and/or location of the default trace file, select the **Change** push button in this dialog box. Doing so allows you to specify a name and/or location for the trace file.

To enable the displayed trace file, click on the **Enabled** push button.

Important: *If you want to return to the default trace file, repeat the above steps to return to the Diagnostic Trace dialog box, and then instead of selecting a file, choose the Cancel button.*

Data packets can be traced when CAICCI/TCP is installed to provide multi-threading support. When the **Snap Packet** box is checked each buffer is displayed as it is sent to or received from the underlying socket layer.

Testing the Configuration

After choosing and configuring a communications protocol, you can test the configuration to see if it has correctly established contact with the host. A series of error messages will be given if the communication fails.

This option is useful because it differentiates between errors related to the communications protocol configuration and errors related to the CA solution application. You can thereby locate the error precisely and quickly, and modify your protocol and/or options until the communication is successful. To test your configuration, choose the Test push button from within the CAICCI-PC Properties dialog box to display the CAICCI Test log.



Click on **Start** to begin the configuration test, the status messages and any error messages will appear in the Test log.

Note: You cannot use the Test option if your CCI settings are not in effect. If you have not already done so, click on OK in the CAICCI-PC Properties dialog box for your settings to take effect. After doing so, re-access the dialog box through the CCI Configurator icon in the Windows/NT Applications Panel, and then click on Test.

Exiting the CAICCI Configurator

After establishing and testing the various communications protocol, close the CAICCI-PC Properties dialog box by clicking OK. The selected protocol and options will remain in effect unless you modify them.

Index

A

Accessible tables view, 3-5, C-9

Accounting information
 passing to CA-IDMS, D-1
 sample code for locating, D-2

Adding data sources, 5-3

Alternate task code, defining, 3-3

API conformance levels
 Core API, A-3
 JDBC driver, B-1
 Level 1, A-4
 Level 2, A-5
 ODBC driver, A-2

ASCII data
 code pages for, 5-17
 translating, 5-15***5-19

C

CA90s Services
 setting up, 3-1
 software requirements, 2-4

CAICCI/PC
 defining to a CA-IDMS system, 3-2
 documentation, 1-5
 installing, 2-3
 interface, 3-1
 mainframe requirements, 2-4
 setup for, 2-5
 use by CA-IDMS Server, 3-1

CA-IDMS
 data type mapping and JDBC, B-3
 data type mapping and ODBC, A-8

 defining, 3-2
 DriverConnect dialog box, 7-2, 7-3
 isolation and lock levels, A-12
 software requirements, 2-4

CA-IDMS Data Source Administrator tool data source, defaults, 5-13

CA-IDMS Server
 Administration dialog box, 5-7
 hardware and software requirements, 2-1
 installation prerequisites, 4-1
 installing, 3-1
 language options, 5-15***5-19

CAIENF facility, 2-4, 3-1

CALC key and SQL access, 3-8

CASERVER task code, 3-3, 4-1

CCI line, defining, 3-2

CCILINE in system generation, 3-2

CECP translation table, 5-18
 using, 5-15***5-19

Character data, translating, 5-15***5-19

Code page
 for host, 5-17
 for PC, 5-17

Communication protocols, supported, 3-1

Conformance levels, A-2, B-1
 API, A-2, B-1

Core API conformance level, A-3
 supported functions, A-3

Core SQL Grammar conformance level, A-6

Country Extended Code Page
 for host languages, 5-17
 for PC languages, 5-17
 using, 5-15***5-19

Creating translation tables, 5-16

D

Data source

- adding, 5-3
- connecting to, 7-1***7-3
- defaults, 5-13
- maintaining, 5-5
- options, 5-12***5-13
- testing connection for, 5-6

Data type mapping, A-8***A-10, A-8, B-3

Data types, CA-IDMS, A-8, B-4

Data, converting

- floating point numbers, C-10
- language options, 5-15***5-19, C-7

Database access

- and page groups, 3-5
- setting up, 3-4

Debugging user sessions, A-1

- error messages, A-1

Default

- data source options, 5-13

Defining

- CA-IDMS host systems, 3-2
- CASERVER task code, 3-3
- CCI line, 3-2
- data sources, using ODBC, 5-2

E

EBCDIC data

- code pages for, 5-17
- translating, 5-15***5-19

Editing translation tables, 5-18

Enable Ensure database option, C-10

Event Notification Facility, 3-1

Extended SQL Grammar conformance level, A-7

EXTERNAL WAIT parameter, on Task statement, 3-4

F

Fetch Real as Double database option, C-10

FILLER record elements, 3-7

FIXED OCCURs element definitions, 3-7

Floating point number, conversion, C-10

Functions

- Core API, A-3
- Level 1 API, A-4
- Level 2 API, A-5

G

Group record elements, 3-7

H

Hardware requirements, personal computer, 2-1

Host code page, 5-17

Hyphens

- in record and set names, 3-6
- in record element names, 3-6

I

INACTIVE INTERVAL parameter, on Task statement, 3-4

Index key and SQL access, 3-8

Installing CA-IDMS Server

- CAICCI/PC, 2-3
- hardware and software requirements, 2-1
- on the mainframe, 3-1
- prerequisites, 4-1
- prerequisites, 2-4

Integrity Enhancement Facility (IEF), A-5

INTERNAL parameter, on Task statement, 3-4

Isolation level, CA-IDMS, A-12

J

JDBC driver

- conformance levels, B-1

data type mapping, B-3
overview of, B-1

L

Level 1 API conformance level, supported functions, A-4

Level 2 API conformance level, supported functions, A-5

Line, CCI, 3-2

Lock level, CA-IDMS, A-12

Log file, C-6
to debug user sessions, A-1
writing trace information to, 5-14

Logging
options, C-6

M

Maintaining data sources, 5-5

Messages, in log file, A-1

Minimum SQL Grammar conformance level, A-5

N

Navigational DML database access
setting up, 3-4
using SQL statements, 3-6

Non-SQL database record
accessing using SQL, 3-6
element name
length, 3-7
transforming, 3-6
name, transforming, 3-6

Non-SQL database set name, transforming, 3-6

O

OCCURS DEPENDING ON record elements, 3-7

ODBC Administrator tool
data source
adding, 5-3
maintaining, 5-5
options, 5-12***5-13
testing connections, 5-6
icon, 4-2

ODBC driver
conformance levels, A-2
data type mapping, A-8
overview of, A-1
supported API functions, A-3, A-4

Options, for data sources
default, 5-13
setting, 5-12***5-13

P

Page groups, and database access, 3-5

Performance considerations
for Cache SQL Tables option, 5-12
for Enable Ensure option, 5-12
for Use Accessible Table View option, 5-12

Personal computer
code pages, 5-17
hardware requirements, 2-1

Protocol
communication, 3-1

Pseudoconversational
processing, A-2

R

Read Committed/Uncommitted option, C-10

READ ONLY/WRITE access mode, C-10

Record
accessing using SQL, 3-6
name, transforming, 3-6

Record element name
length, 3-7
transforming, 3-6

REDEFINES clause
and access by SQL, 3-7
with control keys, 3-8

RHDCNP3S task code, 3-3

Row, bulk fetch, C-10

S

Saving translation tables, 5-19

Server

setting up, 5-7

Set name, transforming, 3-6

Set up

database access, 3-4

SQL to non-SQL database access, 3-5

Set up for

CA90s Services, 3-1

CAICCI/PC, 2-5

CA-IDMS Server on the mainframe, 2-4

CASERVER task, 4-1

servers, 5-7

Setting default data source options, 5-13

Software requirements

CA90s Services, 2-4

CA-IDMS, 2-4

Sort key and SQL access, 3-8

SQL conformance levels

Core SQL Grammar, A-6

Extended SQL Grammar, A-7

Minimum SQL Grammar, A-5

SQL Conformance Levels, A-5, B-3

SQL Data Sources dialog box, 7-2

SQL database access

sample CA-Visual Realia COBOL program, D-1

setting up, 3-4, 3-5

SQL synonym, for non-SQL records, 3-7

SYSCA.ACCESSIBLE_TABLES view, 3-5, C-9

T

Tables, accessible, 3-5, C-9

Task code

defining, 3-3

for set up, 4-1

Testing connections, for ODBC data sources, 5-6

Tracing, session tasks, 5-14

Transaction isolation levels, A-12

Translation editor tool, 5-16

keystrokes, 5-18

Translation table, 5-15***5-19

CECP, 5-15***5-19, 5-18

customizing, 5-18

for non-character-based languages, 5-15***5-19

saving, 5-19

U

Use Accessible Table View option, 3-5, C-9

User session

debugging, A-1

V

View, SYSCA_ACCESSIBLE_TABLES, 3-5, C-9