SYBASE®

Programmer's Reference for Client Services Applications

Mainframe Connect™ Client Option

15.0

IBM CICS, IMS, and MVS

DOCUMENT ID: DC35606-01-1500-02

LAST REVISED: July 2007

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Client Services Applications (CSAs) are customer written applications that enable CICS applications to access local area network (LAN) resources. This book describes how to design, code, and test CSAs.

Audience

This guide is for anyone responsible for:

- Designing, coding, and testing CSAs in one of the supported programming languages (COBOL II, assembler, PL/1, or C)
- Implementing CSAs in CICS
- Administering Open ClientConnect™, Open ServerConnect™, or DirectConnect™
- Administering database management systems
- Supporting data transfer and staging

How to use this book

Each chapter in this book represents a task and each appendix represents reference information to help you accomplish a task. CSA examples are provided in COBOL II.

If you are not familiar with CICS and the CICS control tables, ask your CICS or system programmer to make the required CICS entries.

This book includes the following chapters:

- Chapter 1, “Introduction,” provides general information about the Mainframe Connect™ mainframe access products and an overview of CSAs and how they work.
- Chapter 2, “Designing a CSA,” discusses information to consider before you design a CSA.
- Chapter 3, “Writing a CSA,” explains how to write a CSA.
- Chapter 4, “Compiling and Testing a CSA,” explains how to compile and test a CSA.
- Chapter 5, “Troubleshooting,” explains how to troubleshoot problems in your CSA program.
- Appendix A, “CSA Commands,” lists and explains the CSA commands.

- Appendix B, “CLIENTC2 Sample CSA,” describes a sample CSA that retrieves results and messages to the CSA through an input pipe.
- Appendix C, “CSAINDX Sample CSA,” describes a sample CSA that transfers data from DB2 through DirectConnect to Adaptive Server® Enterprise.
- Appendix D, “CSARESCK Sample CSA,” describes a sample CSA that sends a group of eight INSERT statements in one request buffer to Adaptive Server Enterprise and checks error messages to determine the success of the requests.
- Appendix E, “The SPAREA,” explains how CSAs use SPAREA and includes SPAREA fields and SPAREA definitions.
- Appendix F, “The SQLDA,” describes how CSAs use the SQLDA.
- Appendix G, “Related Products and Documentation by Component,” lists of related Sybase products, related IBM products, and Mainframe Connect™ documentation.

There is also a Glossary provided at the back of the book.

Product name changes

The following table describes new names for products in the 12.6 release of the Mainframe Connect Integrated Product Set.

Old product names	New product name
• Open ClientConnect for CICS • Open ClientCONNECT for CICS	Mainframe Connect Client Option for CICS
• Open ClientConnect for IMS and MVS • Open ClientCONNECT for IMS and MVS	Mainframe Connect Client Option for IMS and MVS
• Open ServerConnect for CICS • Open ServerCONNECT for CICS	Mainframe Connect Server Option for CICS
• Open ServerConnect for IMS and MVS • Open ServerCONNECT for IMS and MVS	Mainframe Connect Server Option for IMS and MVS
• Mainframe Connect for DB2 UDB • Mainframe CONNECT for DB2/MVS-CICS	Mainframe Connect DB2 UDB Option for CICS
• DirectConnect for OS/390 • DirectCONNECT for DB2/MVS	DirectConnect for z/OS

The old product names are used throughout this book, except for on the title page.

Note This book also uses the terms MVS and OS/390 where the newer term z/OS would otherwise be used.

Related documents

The documentation set consists of:

- The *Release Bulletin* for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Product Manuals Web site.

- Mainframe Connect Server Option for CICS *Installation and Administration Guide* – describes configuring the network, installing Open ServerConnect, setting up security, and troubleshooting for an MVS-CICS environment.

- Mainframe Connect Server Option for IMS and MVS *Installation and Administration Guide* – describes configuring the network, setting up APPC communications, installing Open ServerConnect, setting up security, and troubleshooting for an IMS or MVS environment.

- Mainframe Connect Client Option for CICS *Installation and Administration Guide* – describes installing and configuring Open ClientConnect, routing requests to a server, and using Sybase isql. This manual also contains instructions for using the connection router and the mainframe-based isql utility.

- Mainframe Connect Client Option for IMS and MVS *Installation and Administration Guide* – describes installing Open ClientConnect, routing requests to a server, and using Sybase isql. This manual also contains instructions for using mainframe-based isql utility.

- Mainframe Connect DB2 UDB Option for CICS *Installation and Administration Guide* – describes configuring the mainframe, installing Mainframe Connect, setting up security, and troubleshooting for a CICS environment.

- Mainframe Connect DirectConnect for z/OS Option *Installation Guide* – describes installing a DirectConnect server and service libraries.

- Enterprise Connect™ Data Access and Mainframe Connect *Server Administration Guide* for DirectConnect – describes administration of the DirectConnect server. Information about administering specific service libraries and services is provided in other DirectConnect publications.
- Mainframe Connect Client Option *Programmer's Reference for PL/1* – describes writing Open ClientConnect programs that call PL/I Client-Library functions. This guide contains reference pages for Client-Library routines and descriptions of the underlying concepts for PL/I programmers.
- Mainframe Connect Server Option *Programmer's Reference for PL/1* – provides reference material for writing Open ServerConnect programs that call PL/I Gateway-Library functions. This guide contains reference pages for Gateway-Library routines and descriptions of the underlying concepts for PL/I programmers.
- Mainframe Connect Client Option *Programmer's Reference for COBOL* – describes writing Open ClientConnect programs that call COBOL Client-Library functions. This guide contains reference pages for Client-Library routines and descriptions of the underlying concepts for COBOL programmers.
- Mainframe Connect Server Option *Programmer's Reference for COBOL* – provides reference material for writing Open ServerConnect programs that call COBOL Gateway-Library functions. This guide contains reference pages for Gateway-Library routines and descriptions of the underlying concepts for COBOL programmers.
- Mainframe Connect Client Option *Programmer's Reference for C* – describes writing Open ClientConnect programs that call C Client-Library functions. This guide contains reference pages for Client-Library routines and descriptions of the underlying concepts for C programmers.
- Mainframe Connect Server Option *Programmer's Reference for Remote Stored Procedures* – provides information for anyone who designs, codes, and tests remote stored procedures (RSPs).
- Mainframe Connect Client Option *Programmer's Reference for Client Services Applications* – provides information for anyone who designs, codes, and tests client services applications (CSAs).
- Mainframe Connect DirectConnect for z/OS Option *User's Guide for Transaction Router Services* – describes configuring, controlling, and monitoring DirectConnect Transaction Router Service Library, as well as setting up security.

- Mainframe Connect DirectConnect for z/OS Option *User's Guide for DB2 Access Services* (for use with MainframeConnect for DB2 UDB) – describes configuring, controlling, and monitoring DirectConnect for OS/390 Access Service, as well as setting up security.
- Mainframe Connect Client Option and Server Option *Open ClientConnect and Open ServerConnect Messages and Codes* – provides details on messages that mainframe access components return. This guide contains all messages returned by Open ServerConnect and Open ClientConnect. (You may not have all of these products at your site.)

.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click Certification Report.

3 In the Certification Report filter select a product, platform, and timeframe and then click Go.

4 Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

This section describes the syntax and style conventions used in this book.

Note Throughout this book, all references to Adaptive Serve Enterprise also apply to its predecessor, SQL Server. Also, Adaptive Server Enterprise (ASE) and Adaptive Server (AS) are used interchangeably.

The Client Option uses eight-character function names, while other versions of Client-Library use longer names. This book uses the long version of Client-Library names with one exception: the eight-character version is used in syntax statements. For example, CTBCMDPROPS has eleven letters. In the syntax statement, it is written CTBCMDPR, using eight characters. You can use either version in your code.

Table 1 explains syntax conventions used in this book.

Table 1: Syntax conventions

Symbol	Explanation
( )	Parentheses indicate that parentheses are included as part of the command.
{ }	Braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you type the option.
[ ]	Brackets indicate that you can choose one or more of the enclosed options, or none. Do not type the brackets when you type the options.
|	The vertical bar indicates that you can select only one of the options shown. Do not type the bar in your command.
,	The comma indicates that you can choose one or more of the options shown. Separate each choice by using a comma as part of the command.

Table 2 explains style conventions used in this book.

Table 2: Style conventions

This type of information	Looks like this
Gateway-Library function names	TDINIT, TDRESULT
Client-Library function names	CTBINIT, CTBRESULTS

This type of information	Looks like this
Other executables (DB-Library routines, SQL commands) in text	the dbrpcparam routine, a select statement
Directory names, path names, and file names	*/usr/bin directory, interfaces* file
Variables	*n* bytes
Adaptive Server datatypes	datetime, float
Sample code	01 BUFFER PIC S9(9) COMP SYNC. 01 BUFFER PIC X(n).
User input	01 BUFFER PIC X(n)
Client-Library and Gateway-Library function argument names	*BUFFER, RETCODE*
Client-Library function arguments that are input (I) or output (O)	*COMMAND* – (I) *RETCODE* – (O)
Names of objects stored on the mainframe	SYCTSAA5
Symbolic values used with function arguments, properties, and structure fields	CS-UNUSED, FMT-NAME, CS-SV-FATAL
Client-Library property names	CS-PASSWORD, CS-USERNAME
Client-Library and Gateway-Library datatypes	CS-CHAR, TDSCHAR

All other names and terms appear in this typeface.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

The HTML documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/products/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area

CHAPTER 1

Introduction

This chapter provides an overview of the Client Services Applications (CSAs) and defines how CSAs work. It includes the following topics:

- CSA overview
- How does a CSA access and return data?
- Exchanging information between CSAs and Open ClientConnect
- Verifying your environment
- Migration considerations
- Summary of CSA programming tasks

CSA overview

This overview answers the following questions:

- What is a CSA?
- What does a CSA do?
- How does a CSA access and return data?

What is a CSA?

The Client Services feature of CICS allows you to write a Client Services Application (CSA). A CSA is a CICS program that accesses LAN-based data sources, including Adaptive Server Enterprise or any database that can be accessed by a DirectConnect. Results can be returned to the CSA for processing or transferred to another database.

A CSA uses standard CICS services to perform its processing. Examples of CICS services a CSA might use are scheduling and security management programs.

Your CSA can include any request valid in the target database (for example, Transact-SQL® to Adaptive Server Enterprise, DBC/SQL to DBC/1012). This includes both valid SQL and all valid extensions; for example, you can execute Adaptive Server Enterprise stored procedures or DBC/1012 macros.

What does a CSA do?

Customers frequently use CSAs to transfer data between databases, such as DB2 and Adaptive Server Enterprise. For example, you can write a CSA to execute a nightly transfer operation or to periodically poll Adaptive Server Enterprises for changes that were made to the database tables during the day or week.

You can write a CSA to run:

- A SELECT statement against Adaptive Server Enterprise and place the results in DB2, a VSAM dataset, or a temporary storage queue
- A SELECT statement and perform application processing on the result rows
- An UPDATE, INSERT, or other non-SELECT statement against the remote database
- A TRANSFER statement to move data between any two databases. See “Transferring data: about the CSAINDX sample code” on page 61 for more information.

  You can also use a CSA to initiate a transfer of data between two platforms, neither of which is the platform on which the CSA is running. For example, a CSA running under MVS-CICS could transfer data between Adaptive Server Enterprise and DB2 in another MVS region. Transfers require DirectConnect to be running.

How does a CSA access and return data?

As the following figure shows, CSAs:

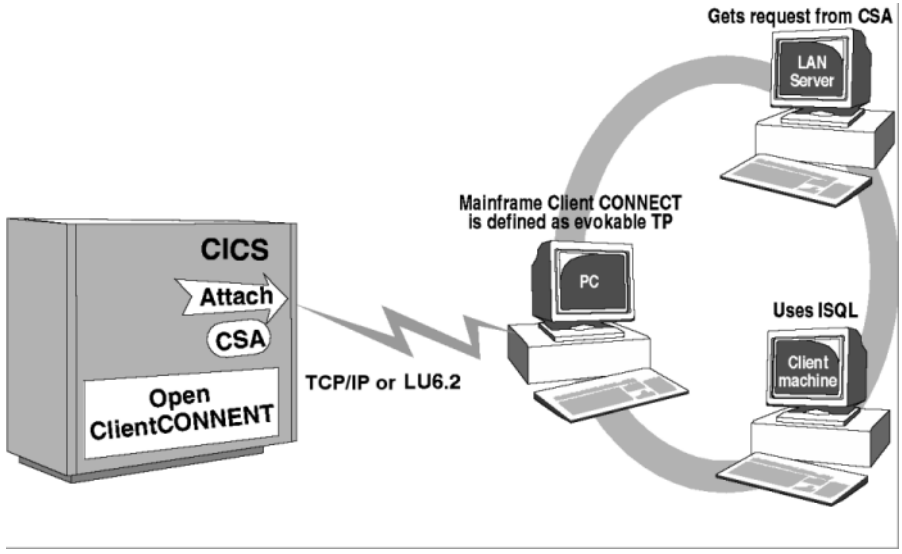
1 Invoke Open ClientConnect.

2 Specify the server connection.

3 Pass the SQL statements through Mainframe Client Connect and to the data source.

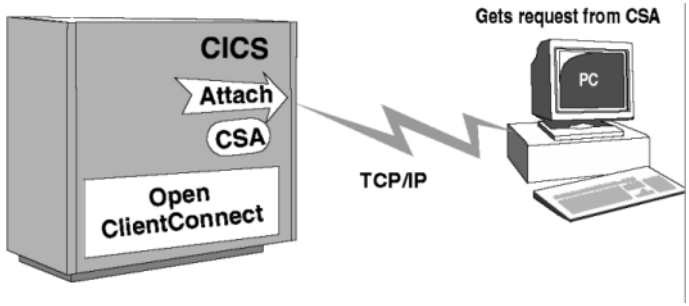
In this way, the CSA acts as a client to the local DBMS using Mainframe Client Connect. Because the CSA is a client, the functions you can perform with a CSA are roughly equivalent to functions you can perform with a client application on the LAN, such as ISQL. See Figure 1-2.

Figure 1-1: CSA processing overview



CSA can go directly to the LAN Server using TCP/IP (Gatewayless). See Figure 1-2.

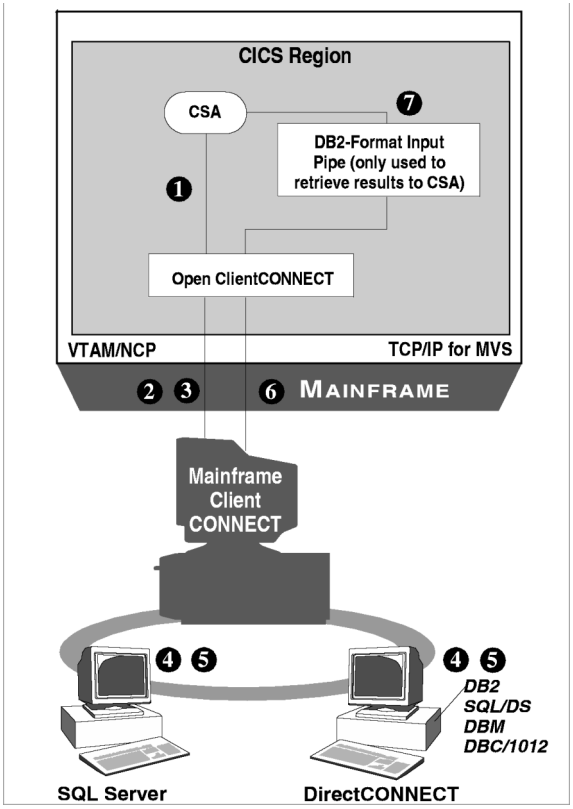
Figure 1-2: CSA processing overview (gateway-less).



CSA processing

A CSA is invoked like any other CICS transaction. The following figure illustrates CSA processing:

Figure 1-3: CSA processing



As Figure 1-3 shows, CSA processing has the following general flow:

1 A CSA calls Open ClientConnect, specifying the server name and the address of the request to be executed at the remote LAN service.

Note The server name is validated in the Open ClientConnect server/connection administrative facilities. The server/connection definitions provide information on the APPC connection and the desired LAN service. See the Mainframe Connect Client Option for CICS *Installation and Administration Guide* for details.

---

2 Open ClientConnect initiates an APPC conversation with Mainframe Client Connect using the connection parameters specified in the server definition.

3 Open ClientConnect sends the request (for example, SQL statements, such as SELECT or TRANSFER statements) to Mainframe Client Connect.

4 Mainframe Client Connect sends the request across the LAN to the target remote database.

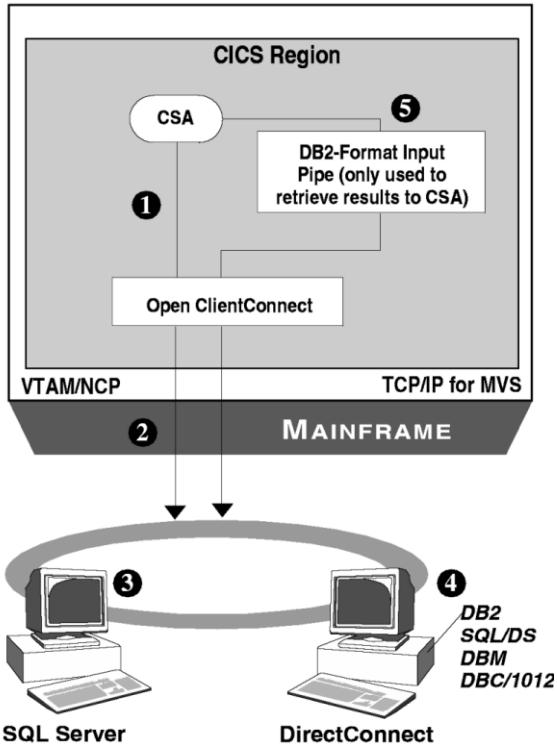
5 The remote database processes the request and returns results to Mainframe Client Connect.

- From an Adaptive Server Enterprise, the results are returned to the CSA directly through Mainframe Client Connect. Therefore, DirectConnect functions, such as SQL translation, are not performed. Only specific, nonconfigurable datatype translation is performed. See your Mainframe Connect DirectConnect for z/OS Option *Installation Guide* (platform specific) for more information.
- From other target databases accessed through a configured DirectConnect, functions, such as datatype conversions, are performed.

6 Mainframe Client Connect sends the results across the APPC link to Open ClientConnect.

7 Using an input pipe, Mainframe Client Connect passes results to the CSA for processing. In the case of a transfer, the results returned to the CSA consist of status information only.

Figure 1-4: CSA Gateway-less processing



As Figure 1-4shows, CSA Gatewayless processing has the following general flow:

1 A CSA calls Open ClientConnect, specifying the server name and the address of the request to be executed at the remote LAN service.

Note The server/connection definitions provide information on the TCP/IP connection and the desired LAN service. See the Mainframe Connect Client Option for CICS *Installation and Administration Guide* for details.

---

2 Open ClientConnect initiates a conversation using the connection parameters specified in the server definition.

3 Open ClientConnect sends the request across the LAN (for example, SQL statements, such as SELECT or TRANSFER statements) directly to Adaptive Server Enterprise or the DirectConnect Server.

4 The remote database processes the request and returns results.

- From an Adaptive Server Enterprise, the results are returned to the CSA directly. Therefore, DirectConnect functions, such as SQL translation, are not performed. Only specific, nonconfigurable datatype translation is performed. See your Mainframe Connect DirectConnect for z/OS Option *Installation Guide* for more information.
- From other target databases accessed through a configured DirectConnect functions, such as datatype conversions, are performed.

5 The results are sent across the TCP/IP link to Open ClientConnect.

Exchanging information between CSAs and Open ClientConnect

Information is exchanged between the CSA and Open ClientConnect using the Stored Procedure Communication Area (SPAREA) and data pipes.

SPAREA

The SPAREA is used to communicate with Open ClientConnect interface.

Data pipes

If your CSA retrieves data, you must open a DB2 format data pipe to Open ClientConnect. (The CSA data pipe can be used only for input.) See Appendix B, “CLIENTC2 Sample CSA” for sample code.

Note If your CSA receives data from an input pipe, you must also read the address of a SQL Descriptor Area (SQLDA) definition from the SPAREA. The SQLDA is a standard data structure that describes the content of the transmitted data records and handles much of the data definition logic. Open ClientConnect places the SQLDA address in the SPAREA. See Appendix F, “The SQLDA” for more information.

System requirements

You use CSAs with Open ClientConnect. Detailed requirements lists are provided in the Mainframe Connect Client Option for CICS *Installation and Administration Guide* using LU 6.2 or TCP/IP.

Host platform

Open ClientConnect must be installed and operational. Your Open ClientConnect administrator must create a server and connection definition. For additional information, see the Mainframe Connect Client Option for CICS *Installation and Administration Guide*.

LAN platform

Mainframe ClientConnect, which is generally installed at the same time as DirectConnect, must be installed and operational with LU 6.2 or TCP/IP. For Gatewayless, Mainframe ClientConnect is not needed.

If the CSA accesses any database other than Adaptive Server Enterprise, the appropriate DirectConnect and database connection must be operational.

If DirectConnect and Mainframe Client Connect reside on the same machine, DirectConnect must be Version 10.5 or later and Mainframe Client Connect must be version 3.1 or later.

Adaptive Server Enterprise

If Adaptive Server Enterprise shares a platform with DirectConnect or Mainframe Client Connect, use SQL Server Version 4.2 or later.

Supported languages

You can write CSAs in any of the four programming languages supported by CICS:

- Assembler
- COBOL II

- PL/I
- C (SAS/C or IBM C/370)

Verifying your environment

After your system administrator installs the prerequisite software and creates an attachment definition, you can use the sample executable CSA AMD2CSP, which is installed with Open ClientConnect, to ensure your environment is set up correctly.

In SPAREA, the server name, ID, and password can be used without an attachment definition.

Note Attachment definitions are only supported for MDI-backward compatibility.

Running the sample CSA

AMD2CSP demonstrates the Client Services for CICS feature and provides testing and troubleshooting functions during Client Services setup. You can use AMD2CSP to test the server definitions, the LU 6.2 or TCP/IP link, and to execute SQL queries in the LAN database.

Note Most installation errors are related to incomplete or incorrect setup of Mainframe Client Connect. However, if you encounter errors between steps 4 and 6 that follow, see your DirectConnect for OS/390 Installation (platform-specific) guide for more information.

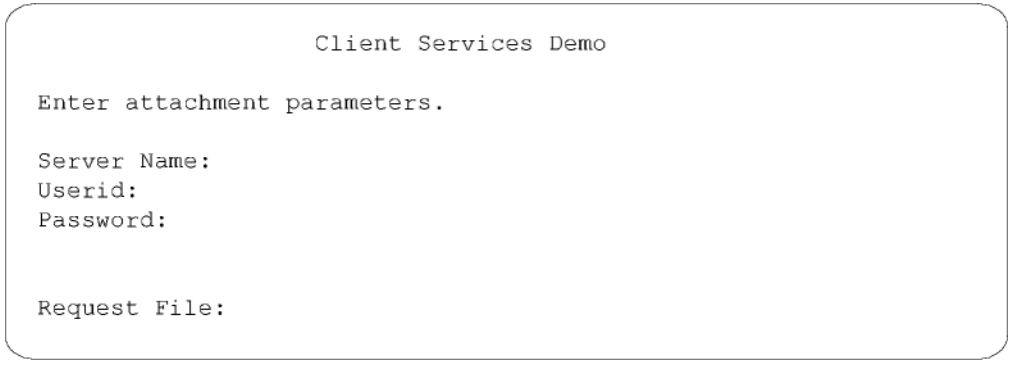
To run the sample CSA (AMD2CSP), perform these steps:

1 Sign on to CICS.

2 Enter the transaction name specified during installation (for example, ASQL), and press Enter.

The following screen appears:

Figure 1-5: Client Services Demo window



Note To terminate AMD2CSP, press Esc, Clear, or any other CICS function key.

3 Enter the necessary attachment properties:

- Server Name is required. Use the name created by the Open ClientConnect Administrator. It can be a valid attachment definition or server name, defined to the OpenClient router tables (LU 6.2) or SYGWHOST macro (tcp/ip.
- Userid and Password are required.
- Request file is optional. It is the name of a temporary storage queue that already contains SQL statements. These statements execute first and the results display before the free-form SQL entry screen appears.

4 Press Enter.

If a request file exists, it is displayed. Go to step 8, for information on reading the results.

If a request file does not exist, the SQL entry screen appears as follows:

Figure 1-6: Enter SQL Requests window

AMD2CSP accepts free-form SQL requests from the screen or from a pre-existing temporary storage queue that you identified as the Request File. Multiple SQL requests can be executed together, and multiple requests can be executed during the transaction.

5 Enter any number of SQL statements on this screen.

You can use multiple statements per line or multiple lines per statement. Delimit the SQL statements by semicolons.

6 Press Enter.

All statements are sent for execution in the remote database, and the following message appears:

```
Processing request.
```

When the requests finish processing, the following message appears:

```
Request finished. <CLEAR> to view results
```

7 Press Clear.

The transaction links to the CEBR program DFHEDFBR to view the results.

A temporary storage queue holds the results in the format in which a typical customer-written CSA would receive it.

For example, integer columns are not translated to displayable characters. You can view non-displayable columns using the CEBR HEX feature (PF2).

If you already processed SQL requests, your new results are appended to the end of the temporary storage queue.

8 View your CSA results in CEBR.

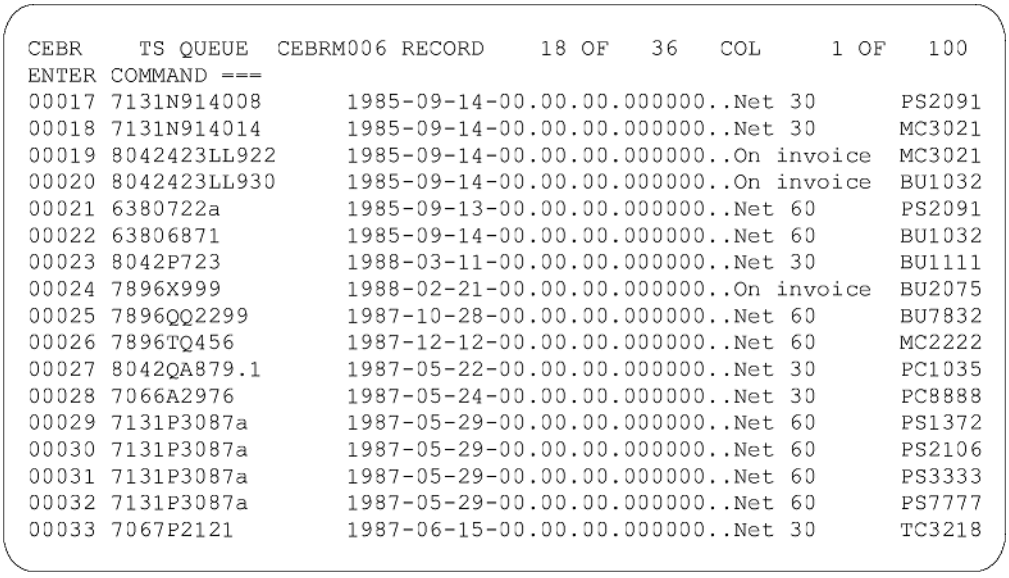
Use the defined PF keys to move through the screens. Figure 1-7, Figure 1-8, and Figure 1-9 show a 36-line temporary storage queue.

Figure 1-7: CEBR output, window 1

```
CEBR    TS QUEUE  CEBRM006 RECORD     1 OF    36   COL    1 OF   100
ENTER COMMAND ===
*************************  TOP OF QUEUE  **********************
00001 LILDEB   will be used for the attachment name.
00002 Changed database context to 'pubs'.
00003 Changed language setting to 'us_english'.
00004 select * from sales;
00005 Number of SQL request rows was 1.
00006 Column Name          Data Type           Columns   Nulls
00007 ____________________________________________________________
00008 stor_id              Character           0001-0004 N
00009 ord_num              Variable Character  0005-0024 N
00010 date                 Time Stamp          0025-0050 N
00011 qty                  Small Integer       0051-0052 N
00012 payterms             Variable Character  0053-0064 N
00013 title_id             Variable Character  0065-0070 N
00014 ____+____1____+____2____+____3____+____4____+____5____+____6____+____7
00015 7066QA7442.3     1985-09-13-00.00.00.000000..On invoice  PS2091
00016 7067D4482        1985-09-14-00.00.00.000000..Net 60      PS2091
```

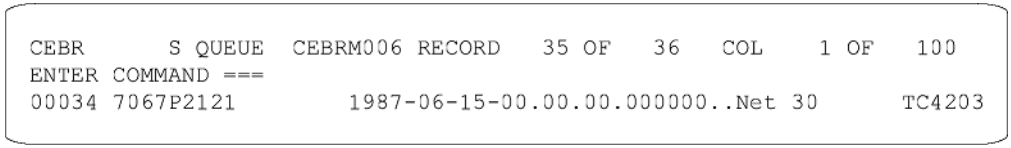
In these figures, the Small Integer field in columns 51–52 does not appear. You can use the PF2: SWITCH HEX/CHAR key to change the view of your results in CEBR. In hex format, you can see the contents of the non-displayable fields.

Figure 1-8: CEBR output, window 2



```
CEBR    TS QUEUE  CEBRM006 RECORD    18 OF   36   COL     1 OF   100
ENTER COMMAND ===
00017 7131N914008      1985-09-14-00.00.00.000000..Net 30      PS2091
00018 7131N914014      1985-09-14-00.00.00.000000..Net 30      MC3021
00019 8042423LL922     1985-09-14-00.00.00.000000..On invoice  MC3021
00020 8042423LL930     1985-09-14-00.00.00.000000..On invoice  BU1032
00021 6380722a         1985-09-13-00.00.00.000000..Net 60      PS2091
00022 63806871         1985-09-14-00.00.00.000000..Net 60      BU1032
00023 8042P723         1988-03-11-00.00.00.000000..Net 30      BU1111
00024 7896X999         1988-02-21-00.00.00.000000..On invoice  BU2075
00025 7896QQ2299       1987-10-28-00.00.00.000000..Net 60      BU7832
00026 7896TQ456        1987-12-12-00.00.00.000000..Net 60      MC2222
00027 8042QA879.1      1987-05-22-00.00.00.000000..Net 30      PC1035
00028 7066A2976        1987-05-24-00.00.00.000000..Net 30      PC8888
00029 7131P3087a       1987-05-29-00.00.00.000000..Net 60      PS1372
00030 7131P3087a       1987-05-29-00.00.00.000000..Net 60      PS2106
00031 7131P3087a       1987-05-29-00.00.00.000000..Net 60      PS3333
00032 7131P3087a       1987-05-29-00.00.00.000000..Net 60      PS7777
00033 7067P2121        1987-06-15-00.00.00.000000..Net 30      TC3218
```

Figure 1-9: CEBR output, window 3



```
CEBR     S QUEUE  CEBRM006 RECORD   35 OF   36   COL    1 OF   100
ENTER COMMAND ===
00034 7067P2121        1987-06-15-00.00.00.000000..Net 30      TC4203
```

9 Use the PF3: TERMINATE BROWSE key to exit CEBR and return to the Enter SQL requests screen (Figure 1-6).

From that screen you can either run another request or exit AMD2CSP.

After exiting from AMD2CSP, you can still view or read the results from the temporary storage queue (TSQ). This queue is named CEBRxxxx, where xxxx is your terminal ID. These results remain in the named queue until the next time AMD2CSP is invoked from that terminal.

Migration considerations

You must recompile existing CSAs with the Open ClientConnect stub programs.

If you are migrating from the Access Server for DB2-CICS Version 2.05.00 or earlier to the Open ClientConnect Version 3.1, you must recompile existing CSAs with the Open ClientConnect Version 3.1 CSA stub programs.

Summary of CSA programming tasks

These are the general steps to build a CSA:

1 Review the design considerations. See Chapter 2, “Designing a CSA.”

2 Prepare a sample CSA as a shell and write the CSA program. See Chapter 3, “Writing a CSA.”

3 Compile, link-edit, and test the CSA in the standard manner for CICS command-level programs. See Chapter 4, “Compiling and Testing a CSA.”

If you encounter problems while processing your completed CSA, see Chapter 5, “Troubleshooting.”

CHAPTER 2

Designing a CSA

CSAs operate in your environment like any other CICS program or function. A CSA can access any CICS program or function that you can access with other programs in the same environment.

This chapter instructs you in how to design your own CSAs and includes the following topics:

- Using CSA commands
- Reviewing the sample CSAs
- Making design decisions
- Understanding DirectConnect property settings
- Understanding how CSAs handle errors

Using CSA commands

This section is a brief introduction to CSA commands. In addition to reading this introductory material, you should review each command in detail before continuing with this chapter. See Appendix A, “CSA Commands” for detailed information about each command.

Use the CSA commands to:

- Establish the client services environment
- Make and close the required LU 6.2 or TCP/IP connections
- Communicate message and status information with Open ClientConnect
- Send SQL requests for execution
- Manage data pipes and receive data from Open ClientConnect

Before you issue a CSA command, move values to the relevant fields in the SPAREA, and then issue a standard system CALL statement.

The following table summarizes the CSA commands and their functions.

Table 2-1: CSA commands and functions

This command	Performs this function	See page
ATTACH	Establishes a logical connection between Open ClientConnect and Mainframe Client Connect and a target DBMS or directly to the target DBMS.	ATTACH on page 44
CLOSPIPE	Closes the data pipe.	CLOSPIPE on page 45
CSSETUP	Establishes the client services environment.	CSSETUP on page 46
DETACH	Terminates the APPC connection.	DETACH on page 46
GETMSG	Retrieves messages generated during remote processing.	GETMSG on page 47
GETPIPE	Reads a record from a data pipe.	GETPIPE on page 47
OPENPIPE	Opens a data pipe.	OPENPIPE on page 48
REQEXEC	Sends the contents of the request buffer for the remote database to execute, and initiates execution.	REQEXEC on page 48
RESCHECK	Checks the status of results returning from the remote database.	RESCHECK on page 49

The CSA commands use the values of fields in the SPAREA as parameters. For details on the SPAREA, see Appendix E, “The SPAREA.”

Using tracing with commands

The SPAREA field SPTRCOPT (see “SPTRCOPT” on page 90) enables or disables tracing. Tracing is off until you turn it on by moving 'Y' to SPTRCOPT. It remains on or off until you change it.

```
MOVE 'Y' TO SPTRCOPT.
 CALL ‘CSSETUP’ USING SPAREA
 ...
 MOVE 'N' TO SPTRCOPT.
```

SPTRCOPT is valid for every command.

Understanding CSA Application Program Interfaces (APIs)

When Open ClientConnect executes a command, it uses the SPAREA SPRC (see "SPRC" on page 88) field to send a return code that indicates the success or failure of the command. If the command is successful, the SPRC field is set to 000. If an error occurs:

1 The SPRC field is set to a 3-digit Open ClientConnect error code. Mainframe Connect Client Option and Server Option *Messages and Codes* contains the Open ClientConnect error codes related to CSAs.

2 The CSA is not allowed to issue any more commands. The CSA should perform any termination processing and then return control to Open ClientConnect.

The following COBOL II statements show an example of return code checking after issuing an OPENPIPE command:

```
CALL 'OPENPIPE' USING SPAREA.
 IF SPRC NOT EQUAL '000' THEN GO TO PERFORM-TERMINATE.
```

In addition to '000', the SPRC field can contain a valid return code of EOF, indicating End Of File on input data.

Reviewing the sample CSAs

Now that you reviewed the information in this chapter on commands as well as Appendix A, "CSA Commands," you are ready to review a sample CSA. The following sample CSA programs are distributed with the Open ClientConnect API (each sample CSA is written in COBOL II):

- CLIENTC2 Accesses Adaptive Server Enterprise, retrieving results and messages to the CSA through an input pipe. See Appendix B, "CLIENTC2 Sample CSA," for a reproduction of the sample.
- CSAINDX Transfers data from DB2 through DirectConnect to Adaptive Server Enterprise. See Appendix C, "CSAINDX Sample CSA," for a reproduction of the sample.
- CSARESCK Sends a group of eight INSERT statements in one request buffer to Adaptive Server Enterprise and checks error messages to determine the success of the requests. See Appendix D, "CSARESCK Sample CSA," for a reproduction of the sample.

Making design decisions

Before you write a CSA, you have several decisions to make and environmental issues to consider:

- What functions will the CSA perform?
- Will the CSA access Adaptive Server Enterprise, remote databases, or both?
- What kind of SQL transformation and datatype conversions will be applied?
- Will the CSA access temporary storage or transient data queues?
- Will the CSA link to other programs or functions?
- What kind of error handling will the CSA require?

Choosing CSA functions

According to your users' requirements, decide what functions the CSA will perform. For example, your CSA can:

- Transfer data between two sources through Mainframe Client Connect. A CSA running on MVS/CICS can transfer data between Adaptive Server Enterprise and any DirectConnect target database, or between any two DirectConnect target databases.
- Access DB2 data, statically or dynamically.
- Access other relational data sources (for example, ADABAS), statically or dynamically.
- Access nonrelational data (for example, VSAM and IMS).
- Invoke other CICS programs.
- Schedule other CICS tasks for execution.

Accessing databases

Your CSA can access any database you have in your CICS environment. For example, it can access:

- MVS data directly

- MVS data indirectly (through DirectConnect)
- Non-MVS data directly (through Adaptive Server Enterprise)
- Non-MVS data indirectly (through DirectConnect)

Understanding SQL and data transformation

SQL transformation depends on whether your CSA accesses a database through a DirectConnect. See “Understanding DirectConnect property settings” on page 24 for more information.

Adaptive Server Enterprise

Mainframe Client Connect does not perform SQL transformation. Therefore, if the target LAN service of your CSA is Adaptive Server Enterprise, the request must be written in Transact-SQL.

Databases accessed through DirectConnect

If the target LAN service of your CSA is a DirectConnect instance, SQL transformation is performed as configured through a property setting on DirectConnect.

The CSA must send requests in SQL appropriate for the transformation level set in the current DirectConnect configuration. Otherwise, before the CSA sends the request it must send a SET statement to set the correct transformation level. For information about SET statements, see the Mainframe Connect DirectConnect for z/OS Option *User's Guide for DB2 Access Services* appropriate for your environment.

Similarly, any data that a CSA retrieves through a DirectConnect instance is converted to Adaptive Server Enterprise format as specified by the current transformation settings. The CSA can send SET statements to define the desired conversion settings.

Accessing temporary storage/transient data queues

You access temporary storage or transient data queues with CSAs the same way you access them with any other program in CICS. For more information on accessing temporary storage or transient data queues with CICS programs, refer to CICS documentation.

Transmitting data to a CSA

All data transmitted between the CSA, Open ClientConnect, Mainframe Client Connect, and the target is sent in Tabular Data Stream™ (TDS) format, which replaces Integrated Exchange Format (IXF). TDS is a Sybase proprietary format, which manages data formatting for you. If the data is coming from Adaptive Server Enterprise, Mainframe Client Connect translates the data into TDS format.

Using data pipes

CSAs use data pipes only to receive data from Adaptive Server Enterprise or DirectConnect. You do not need to define data pipes if you are only transferring data between data sources or sending SQL commands other than SELECT.

You define the type of pipe (input) and the format of the data being transmitted using the CSA commands described in Appendix A, “CSA Commands.” The data pipe management commands are OPENPIPE, GETPIPE, and CLOSPIPE.

Input pipes

The CSA uses input pipes to read rows of data retrieved from a database or LAN-based service. This COBOL II example shows a CSA opening, reading from, and closing a pipe:

```
MOVE 'INPUT' TO SPMODE      – defines an input pipe
 MOVE 'DB2' TO SPFORMAT.    – defines input pipe as DB2 format
 CALL OPENPIPE USING        – opens the pipe
SPAREA.
 CALL GETPIPE USING         – reads from the pipe
SPAREA.
 CALL CLOSPIPE USING        – closes the pipe
SPAREA.
 CALL GETMSG USING          – gets messages from the local platform
SPAREA.
```

Defining data format for input pipes

When defining an input data pipe, you must specify the format of the data to be transmitted through the pipe.

Warning! Always define a CSA input pipe as DB2 format, regardless of the data source targeted.

The DB2 format causes Open ClientConnect to set the address of a SQL Descriptor Area (SQLDA) definition when it returns data to the CSA.

The SQLDA is an IBM-standard data structure that describes the content of the transmitted data records and handles much of the data definition logic. See “Using input pipes: about the CLIENTC2 sample code” on page 51 for an example of a CSA that reads a SQLDA definition. See Appendix F, “The SQLDA” for more information. In addition, see IBM DB2 SQL Reference for information about the SQLDA.

Linking to other programs

When you link to or call another program from a CSA, you must use a format that allows the program to return processing control to the CSA. If the program does not return control to the CSA (for example, with an XCTL), CICS makes a copy of the SPAREA for the called program instead of pointing to the original SPAREA. Since control was not returned to CSA the results are unpredictable.

To avoid this, use one of these commands to link to another program:

```
CICS LINK programname
CALL programname
```

Handling errors

The CSA must include code to handle errors it receives from Open ClientConnect and, optionally, from DB2 or any other database it accesses.

Errors from Mainframe Client Connect are recorded in the *SPRC* (see “SPRC” on page 88) field of the SPAREA. Your CSA code should check that field for errors after issuing any CSA command.

See “Understanding how CSAs handle errors” on page 24 for examples of error handling in CSAs. See Mainframe Connect Client Option and Server Option *Messages and Codes* for error codes relevant to CSAs.

Accessing databases

With CSAs, the Mainframe Client Connect receives SQL statements and TRANSFER statements (or other statements valid in the target database) from Open ClientConnect. TRANSFER statements are processed by DirectConnect, while other statement types are passed along to the target database for processing there.

Understanding DirectConnect property settings

If your CSAs will access remote databases through DirectConnect, you need to be aware of how the DirectConnect configuration properties affect request and result processing.

Adaptive Server Enterprise uses Transact-SQL, while DB2 uses IBM’s version of SQL. Consequently, SQL statements written for Adaptive Server Enterprise generally do not perform as expected when executed against DB2. To handle this discrepancy, DirectConnect can be configured one of the following ways:

- PASSTHROUGH mode, to pass through the IBM SQL
- SYBASE mode, to transform the IBM SQL to Transact-SQL

For more information on DirectConnect property settings, see the appropriate Mainframe Connect DirectConnect for z/OS Option *User's Guide for DB2 Access Services*.

Understanding how CSAs handle errors

This section explains how CSAs use the server name, the SPAREA, the CSA commands, the SQLDA, and the SPAREA message fields for error handling.

Using a server name

All CSAs must include a reference to a server name previously created with the Open ClientConnect Administration menu. This program includes the server name after the third comment in:

```
WORKING-STORAGE SECTION:
01 ATTACH-NAME PIC X(32) VALUE 'SQLSERVE'.
```

The server name includes information, such as the CICS Connection Name, the name of the Adaptive Server Enterprise or DirectConnect that processes the request, and the Mainframe Client Connect name. See your Open ServerConnect and Open ClientConnect installation guides for more information.

Using the SPAREA

The SPAREA is the storage area that passes information between the CSA and Open ClientConnect. The CSA supplies the SPAREA and notifies Open ClientConnect that it is available using the CSSETUP command. The statement copying the SPAREA definition into the CSA must be in the LINKAGE SECTION so that the SPAREA is available to Open ClientConnect:

```
01  STORE-PROC-AREA.
    COPY SPAREAC.
```

Note The CSA allocates the storage for the SPAREA in the LINKAGE SECTION. (The programmer is responsible.)

The following example illustrates how to use the SPAREA to pass information to Open ClientConnect about the data pipe the CSA is using:

```
MOVE 'INPUT '          TO SPMODE.
 MOVE 'DB2'             TO SPFORMAT.
 CALL 'OPENPIPE' USING SPAREA.
```

Two SPAREA fields are used with the OPENPIPE command. The SPMODE field specifies the mode (always input for CSAs) of the data pipe, and the SPFORMAT field specifies the format (always DB2 for CSAs) of the data to be transmitted through the pipe.

See Appendix E, “The SPAREA” for detail on all the SPAREA fields and their uses.

Using CSSETUP

After you copy an SPAREA into your CSA, issue the CSSETUP command to inform Open ClientConnect that a CSA is ready to process and that the CSA has an SPAREA definition to use for communication. CSSETUP initializes the SPAREA structure.

Using CSA commands

You can invoke the CSA commands with a standard CALL statement in all the supported programming languages. The CSA commands used in CLIENTC2 are ATTACH, CSSETUP, REQEXEC, RESCHECK, GETMSG, OPENPIPE, GETPIPE, and DETACH.

Understanding CSA command formats

In COBOL II, the CSA command should be enclosed in quotes; in the other supported languages, quotes are not necessary. COBOL II handles commands like this:

```
CALL 'REQEXEC' USING SPAREA.
```

Note The REQEXEC call in COBOL is a static call. The REQEXEC module is linked into the CSA load module. The quotes make the call a static call.

Using RESCHECK and GETMSG

You should always include RESCHECK and GETMSG commands after the REQEXEC to ensure that you receive all error messages generated during processing. RESCHECK determines whether or not there are messages and result rows (OPENPIPE and GETPIPE retrieve the result rows), and GETMSG retrieves the messages.

For more information on the CSA commands, their formats and results, see Appendix A, “CSA Commands.”

Using the SQLDA

The SQLDA is a structure of pointers and values that provide data structure information about data being transmitted in multi-column rows. The SQLDA definition passed to the CSA provides metadata about the data being transmitted.

Note The SQLDA is an IBM standard; see *IBM DB2 SQL Reference* for more information.

With CSAs that receive data through an input pipe, you set up a SQLDA template in the program and then access the address of the actual SQLDA that Open ClientConnect sends. See Appendix F, “The SQLDA” for more explanation.

Specifying error handling

The four SPAREA fields used to communicate status and messages between the CSA and Open ClientConnect are described in the following table and in the sections following the table.

Note For the size of the datatype fields, see the “SPAREAC COBOL II definition” on page 92.

Table 2-2: SPAREA error handling fields

SPAREA field	Use
SPRC	Stored Procedure (SP) return code. Open ClientConnect indicates the success or failure of a CSA command in this field. Possible values are: • '000' indicates successful completion • *'xxx'* indicates an Open ClientConnect error message • 'EOF' indicates End of File on input data • 'ACE' indicates APPC Communication Error (when the Open ClientConnect configuration property, Temporary Storage Type, is set to None)
SPSTATUS	SP status. Open ClientConnect indicates the success or failure of processing in the remote database in this field. Possible values are: • 'OK' indicates success • 'E' indicates error • 'W' indicates warning • 'R' results are available • Blank indicates no results available
SPIND	SP indicator. Open ClientConnect indicates the availability of an additional message. Possible values are: • 'M' indicates messages are available • [Blank] indicates messages are not available
SPMSG	SP message. Open ClientConnect communicates message text to the CSA using this field.
SPCODE	SP code. No longer used.

Open ClientConnect error messages related to CSAs are listed in Mainframe Connect Client Option and Server Option *Messages and Codes*.

Using the SPRC field

Open ClientConnect uses the SPRC field in the SPAREA to communicate the success or failure of a CSA command.

Note Always include a check in your code for the SPRC field after issuing any CSA command.

These code fragments show how a CSA accesses the SPRC (return code) field to get this information:

```
CALL 'DETACH' USING SPAREA.
IF SPRC NOT = '000'
     MOVE 'DETACH'      TO WS-STUB-NAME
     PERFORM 6900-STUB-CALL-ERROR THRU 6900-EXIT
     GO TO 0000-GET-OUT-NOW.
MOVE SPRC             TO WS-STUB-SPRC.
 MOVE SPMSG         TO WS-STUB-SPMSG.
EXEC CICS
     WRITEQ TS QUEUE('CSEXQUE')
         FROM(WS-STUB-ERROR-MSG) NOSUSPEND
     RESP(CICSRC)
 END-EXEC.
```

Using the SPSTATUS field

Open ClientConnect uses the SPAREA SPSTATUS field to communicate the success or failure of processing in the remote database. The CSA must check the status of database processing, as this code fragment illustrates:

```
IF SPSTATUS NOT = (spaces)
     MOVE 'N'            TO WS-RESCHECK-DONE-SW
     ADD 1               TO SPSTATUS-CNT
     MOVE SPSTATUS-CNT   TO WS-RESCHECK-NUMBER
     MOVE SPSTATUS       TO WS-SPSTATUS-OUT
     MOVE SPCODE         TO WS-SPCODE-OUT
     MOVE SPIND          TO WS-SPIND-OUT
     EXEC CICS
         WRITEQ TS QUEUE('CSEXQUE')
             FROM(WS-RESCHECK-MSG) NOSUSPEND
             RESP(CICSRC)
     END-EXEC
```

In this example, the CSA places relevant error information in a temporary storage queue for later examination.

Using the SPIND field

Open ClientConnect uses the SPAREA SPIND field to inform the CSA that an additional message is available. The code fragment shows how a CSA accesses the SPIND field to get this information:

```
IF SPIND = 'M'
     PERFORM 6100-GET-MESSAGES    THRU 6100-EXIT
         UNTIL SPIND NOT = 'M'
 END-IF.
```

Using the SPMSG field

Open ClientConnect communicates message text to the CSA using the SPMSG field. In the sample program, issuing the GETMSG command retrieves the message text from *SPMSG*; the messages are then written to temporary storage:

```
CALL 'GETMSG' USING SPAREA
 IF SPMSG NOT = SPACES
     EXEC CICS
         WRITEQ TS QUEUE('CSEXQUE')
             FROM(SPMSG) NOSUSPEND
         RESP(CICSRC)
     END-EXEC
 END-IF.
```

See Appendix A, “CSA Commands” for details about the MESSAGE command and Appendix E, “The SPAREA” for information about the SPAREA.

CHAPTER 3

Writing a CSA

After you gather requirements for and design your CSA, you are ready to write the CSA, using a sample CSA as a base.

This chapter contains the following topics:

- Choosing a sample CSA
- Renaming the sample
- Making the CICS resource table entries
- Creating the server and connection definitions
- Testing the sample
- Writing the CSA

Choosing a sample CSA

Sybase recommends that you select a sample CSA in the programming language you are using as a shell for your application. The sample CSAs are provided with the Open ClientConnect.

The following table lists the sample programs and definitions available to you:

Table 3-1: Samples included on the Open ClientConnect

Sample	Description
CLIENTC2	Accesses Adaptive Server Enterprise, retrieving results and messages to the CSA through an input pipe. This sample program is written in COBOL II.
CSAINDX	Transfers data from DB2 through DirectConnect to Adaptive Server Enterprise. This sample program is written in COBOL II.
CSARESCK	Sends a group of eight INSERT statements in one request buffer to Adaptive Server Enterprise and checks error messages to determine the success of the requests. This sample program is written in COBOL II.

Renaming the sample

After you select a sample CSA to use as a shell, rename the sample using the naming conventions of standard mainframe programs used at your site for the CSA name.

Making the CICS resource table entries

Make entries for your CSA in your CICS PPT and PCT so you can test it. Generally, the Systems Programmer makes the entry for the CSA in the CICS PPT and PCT manually using RDO.

In addition, if the CSA accesses DB2 directly—not through a DirectConnect—you also need to make an RCT entry.

Creating the server and connection definitions

You create connection definitions using the Open ClientConnect utilities.

- For LU 6.2, use SYMS and SYMC transactions.
- For TCP/IP, use SYGWHOSTS in Open Client Customization module.

Note In the Customization Module (SYCTCUST, a JCL member) there is a default protocol field for IBM TCP/IP, Interlink TCP/IP, LU 6.2 and CPIC.

Testing the sample

Before you begin to write your CSA, test the sample you are using as a shell. You can test your client requests (the commands your CSA sends to the LAN server) with a LAN-based client application such as ISQL and ASQL. This helps isolate SQL request issues from any connectivity issues

If you need detailed instructions on testing the sample, go to Chapter 4, "Compiling and Testing a CSA."

Writing the CSA

This section describes how to prepare to write the CSA and lists some of the programming tasks.

Before you write the CSA

To prepare to write your CSA:

- Ensure your environment is operational.
- Ensure your connectivity is operational by setting up a client application, such as ISQL or ASQL on the same machine as Mainframe Client Connect. Use it to:
  - Connect to the target server to isolate any network connectivity issues using ASQL
  - Test SQL and CSA commands to isolate any command syntax issues using ASQL
- Select the SPAREA definition for the programming language you are using. Appendix E, "The SPAREA" includes a list of SPAREA definitions.
- If you did not already do so, create a server definition to use during testing. You create a server definition using the Open ClientConnect administration facilities. See the Mainframe Connect Client Option for CICS *Installation and Administration Guide* for information.
- Make the necessary CICS resource table entries for your program.

Write the CSA

You can use the database access code from existing programs in your CSA. Whenever possible, use existing data definitions or code from other programs. We recommend that you use one of the sample CSA programs and SPAREA definitions provided with the Open ClientConnect API as a base for your own CSA.

CSA programming tasks include:

1 Define input data pipes if the CSA retrieves results.

2 Use the CSA commands, such as GETMSG and RESCHECK, whenever appropriate. Appendix A, “CSA Commands” provides detailed information on commands.

3 Read from and writing to the SPAREA that the CSA shares with Open ClientConnect.

4 Specify error handling.

CHAPTER 4

Compiling and Testing a CSA

This chapter describes how to compile and test your CSAs and includes the following topics:

- Before compiling the CSA
- Compiling a CSA
- Testing a CSA

Before compiling the CSA

To test the CSA, first verify that your attachment definition is working. You can do this by running the sample CSA Transaction ASQL (AMD2CSP program name) and specifying your own attachment definition. See “Running the sample CSA” on page 10 for details.

After you confirm that you can connect to the remote DBMS, verify that your CSA is working using your normal application testing methods.

Note The sample JCL is provided with the Open ClientConnect API.

Compiling a CSA

Sybase supplies the CSA stub programs as both load modules and object code, so use the example appropriate for your site. The stub libraries are provided on the Open ClientConnect API.

Compile and link-edit the CSA source code using the standard methods for command-level CICS programs.

Note Each time you link-edit, you must also perform a CICS NEWCOPY.

Note The Open ClientConnect CSA stub library must be included in the SYSLIB concatenation sequence of the LNKEDT step. The CSA must be linked so it runs in extended storage (that is, AMODE(31) RMODE(ANY)).

CSAs must be linked above the 16MB line in 31-bit addressing mode. To do this, include a JCL statement similar to the following:

```
//LNKEDT   EXEC PGM=IEWL,PARM='parms AMODE(31)
                RMODE(ANY)'
```

The concatenation sequence for SYSLIB in the link-edit step must include a data definition (DD) statement for the stub library, either in load module format or in object code format.

LOAD module format

If you want to link the load modules, use this SYSLIB statement in your JCL:

```
//SYSLIB   DD  DSN=SYBASE.0CSA310B.CICS.LOADLIB,
               DISP=SHR
```

Object code format

If you want to compile one of the sample programs provided with Open ClientConnect, use this SYSLIB statement in your JCL:

```
//SYSLIB   DD  DSN=SYBASE.0CSA310B.CICS.OBJLIB,
               DISP=SHR
```

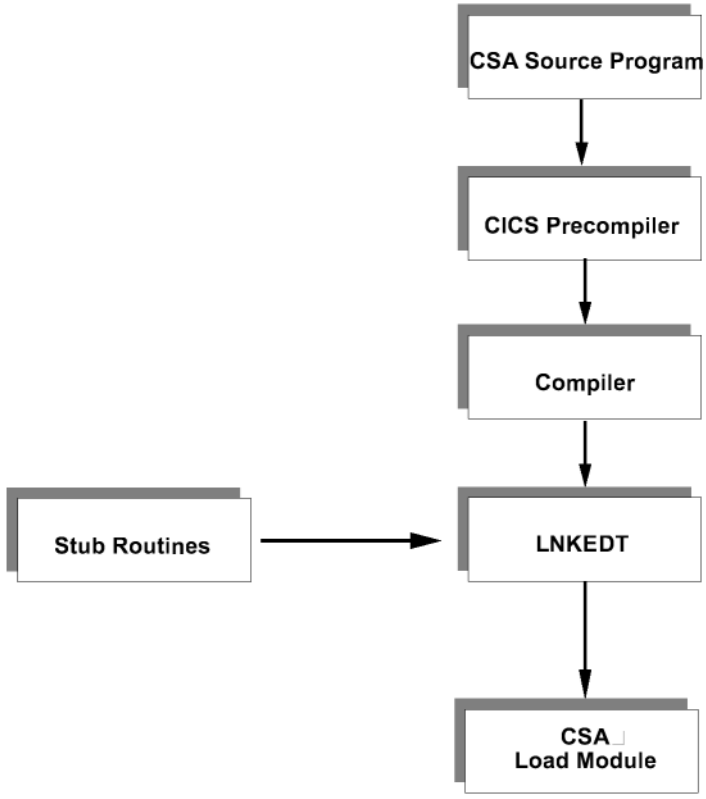
See your Mainframe Connect Client Option *Installation and Administration Guide* for IBM for the correct DSN for this release of the software.

Understanding the linkage

During the link-edit, stub routines are included in the resulting load module for the CSA. The stub routines provide the linkage between the CSA and Open ClientConnect.

Figure 4-1 illustrates the process involved.

Figure 4-1: Compiling a CSA without DB2

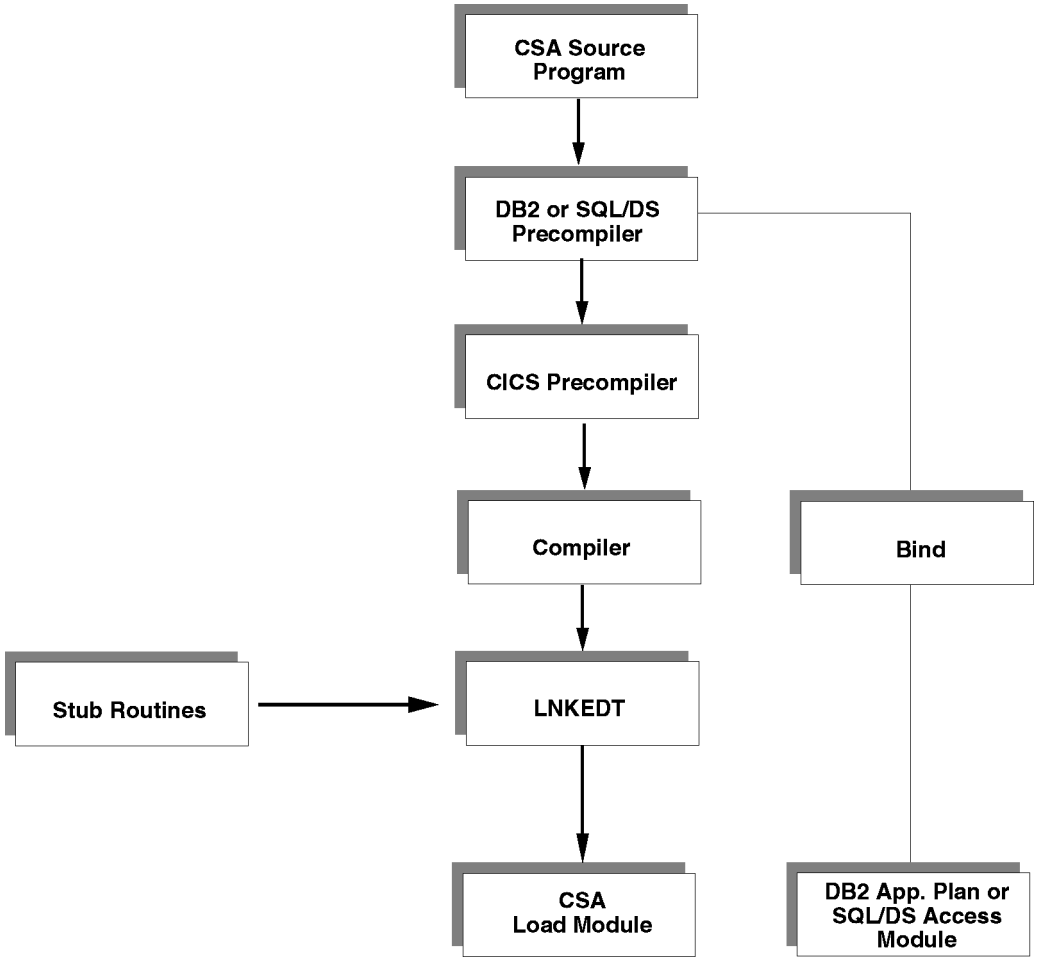


Accessing DB2

If your CSA accesses DB2, there is an additional precompile process. The DB2 precompiler produces an application plan that must be bound to DB2. In addition, the Systems Administrator must grant all CSA users EXECUTE authority on the plan.

Figure 4-2 illustrates the process involved.

Figure 4-2: Compiling a CSA with database access



Additionally, if your CSA accesses DB2 directly (not through DirectConnect), your System Administrator must make a CICS RCT entry for the CSA transaction.

Testing a CSA

At this point, you are ready to test your CSA.You should test your CSA in the same manner as you test any CICS command-level program.

CHAPTER 5

Troubleshooting

This chapter describes how to troubleshoot problems in your CSA program and discusses the following topics:

- Using CSA debugging tools
- Resolving common connectivity problems
- Resolving common coding problems
- Using the Mainframe Client Connect log

Using CSA debugging tools

You have several debugging tools available:

- ISQL and CICS on the LAN
- ASQL
- Open ClientConnect traces
- MainframeConnect traces
- CICS CEDF transaction
- Third-party debugging tools

ISQL and CICS

To verify you have the necessary client network setup on the LAN server that Open ClientConnect communicates with, always test the SQL requests with ISQL from CICS.

ASQL

To verify that the ATTACH connection, the LAN server, and the SQL statement itself are all valid, you can test the SQL requests in ASQL before you include them in your CSA source code.

Open ClientConnect traces

You can use the Open ClientConnect traces to verify that connections are made, requests are sent, and messages and data are received.

You control the internal tracing in your CSA code. When the SPTRCOPT field contains 'Y' and a command is issued, internal trace records write to the TSQ, CE*xxxxxx*, where *xxxxxx* is the first six characters of the user ID. The trace records show the internal processing flow and return codes from all Open Client calls.

Mainframe Client Connect traces (MCC)

Use Mainframe Client Connect traces for additional debugging, if the problem is relevant to the LAN server.

CICS CEDF transaction

Use the CICS CEDF transaction to debug CICS issues or to verify the logic path of your CSA.

Third-party debugging tools

Use third-party debugging tools, such as Intertest, for additional debugging.

Resolving common connectivity problems

The most common errors are 084 and 086 in Open ClientConnect. These errors imply that LU 6.2 or TCP/IP cannot get the connection it requires to send the request. There are three possible causes:

- Open ClientConnect or Mainframe Client Connect is not set up correctly.

  Verify that Open ClientConnect or Mainframe Client Connect was set up correctly.

- Open ClientConnect or Mainframe Client Connect is not configured to the communications server as a transaction program that can be invoked.

  See your Open ClientConnect or Mainframe Client Connect installation guides.

- The connection to the LAN Server is broken; a bridge or router is involved. Contact your network support staff.

A CSA can receive the following message or some other connectivity-related error message:

```
SQL Server unavailable or does not exist
```

To isolate the problem, perform these diagnostic actions:

1 First, verify that the LAN Server can be accessed by using isql on the mainframe to log on.

2 Verify that the Mainframe Client Connect machine has the client network DLLs it needs by using ISQL on the same machine to log on to the LAN Server.

3 Verify that the *SERVICENAME* in the attach record is valid.

4 Check the SYGWHOST macros (in Open Client customization) port number and IP address.

Resolving common coding problems

CSAs use only DB2 input pipes. A CSA cannot use output pipes. You must use INSERT and UPDATE commands to send data to the LAN server.

When you batch SQL commands to Adaptive Server Enterprise, you can scroll through messages with the GETMSG command. However, if Adaptive Server Enterprise does a ROLLBACK, the GETMSG command results in only one message.

Using the Mainframe Client Connect log

Frequently, you can trace initial problems running CSAs to incorrect setup of Mainframe Client Connect, which is usually installed on the same machine as DirectConnect. See the Mainframe Connect DB2 UDB Option for CICS *Installation and Administration Guide* for more information.

APPENDIX A

CSA Commands

This appendix discusses the following topics:

- Command examples
- Commands

Command examples

These examples show how each programming language opens a DB2 format input pipe.

Assembler language example

```
MVC SPMODE,=CL6'INPUT'
 MVC SPFORMAT,=C'DB2'
 CALL OPENPIPE,SPAREA
```

COBOL II language example

```
MOVE 'INPUT' TO SPMODE.
 MOVE 'DB2' TO SPFORMAT.
 CALL 'OPENPIPE' USING SPAREA.
```

PL/I language example

```
SPMODE='INPUT';
 SPFORMAT='DB2';
 CALL OPENPIPE(SPAREA);
```

C language example

```
memcpy(spPointer->>spmode, "INPUT ",
 sizeof(spPointer->>spmode));
 memcpy(spPointer->>spformat, "DB2",
 sizeof(spPointer->>spformat));
 openpipe(spPointer);
```

Note All the other examples in this chapter are in COBOL II.

Commands

The following CSA commands are explained in this appendix:

ATTACH

Description — Establishes a logical connection from Open ClientConnect to Mainframe Client Connect and, ultimately, to a target DBMS using the SNA or TCP/IP link.

Syntax — Syntax varies with the programming language.

Parameters

The ATTACH command uses values from these SPAREA fields:

- *SPSERVER* (see "SPSERVER" on page 89) specifies the name of an entry in the Open ClientConnect router/server table.
- *SPUSERID* (see "SPUSERID" on page 89) specifies the user ID required to access the target database.
- *SPPWD* (see "SPPWD" on page 89) specifies the password required to access the target database.
- *SPATTACH* (see "SPATTACH" on page 89) specifies Attachment file name, for backward compatibility to MDI.

Examples

COBOL II

```
MOVE 'SPDB2N1' TO SPSERVER.
MOVE 'JWILS32' TO SPUSERID.
MOVE '3LTB2AP' TO SPPWD.
CALL 'ATTACH' USING SPAREA.
```

Usage

During the attach process, Open ClientConnect establishes an APPC conversation with Mainframe Client Connect, which in turn, establishes a conversation with the target DBMS, using a LAN protocol.

Before it calls the ATTACH command, the CSA must:

- Provide a router/server name
- Move the user ID and password to the SPAREA

CLOSPIPE

Description

Closes a pipe.

Syntax

Syntax varies with the programming language.

Parameters

The CLOSPIPE command uses the value from the SPAREA field SPMODE (see page "SPMODE" on page 88) that specifies whether the data pipe is opened for input.

Examples

COBOL II

```
MOVE 'INPUT' TO SPMODE.
CALL 'OPENPIPE' USING SPAREA.
...
CALL 'CLOSPIPE' USING SPAREA.
```

Usage

- CSA data pipes are always input mode and DB2 format.

- Use the CLOSPIPE command after the OPENPIPE command was issued. The CSA must issue the CLOSPIPE command before invoking the DETACH or the next REQEXEC command.

CSSETUP

Description

Establishes the client services environment and initializes many fields in the SPAREA. This must be the first CSA command that the CSA issues.

Syntax

Syntax varies with the programming language.

Examples

COBOL II

1 Issue a CSSETUP command:

```
CALL 'CSSETUP' USING SPAREA.
```

2 Turn on tracing before you call CSSETUP:

```
MOVE 'Y' TO SPTRCOPT.
CALL 'CSSETUP' USING SPAREA.
```

Usage

DETACH

Description

Closes the connection between Open ClientConnect and the Mainframe Client Connect server.

Syntax

Syntax varies with the programming language.

Examples

COBOL II

```
CALL 'DETACH' USING SPAREA.
```

Usage

GETMSG

Description — Retrieves messages that DirectConnect or Adaptive Server Enterprise generate during SQL request processing.

Syntax — Syntax varies with the programming language.

Examples — **COBOL II**

1 Test *SPIND* before issuing a GETMSG:

```
PERFORM UNTIL (SPIND NOT = 'M')
     CALL 'GETMSG' USING SPAREA
     PERFORM PROCESS-MSG
 END-PERFORM.
```

Usage

- When the GETMSG command runs, Open ClientConnect supplies the next available message in the *SPMSG* (see page "SPMSG" on page 89) field in the SPAREA. If a message is not available, the *SPIND* (see page "SPIND" on page 89) and *SPMSG* fields are set to blanks. The CSA should continue to issue GETMSG commands until all messages are retrieved.
- Before issuing this command, you can determine whether you received messages by examining the *SPIND* field in the SPAREA. *SPIND* contains one of the following values:
  - 'M' indicates that additional messages are available.
  - [Blank] indicates that additional messages are not available.

GETPIPE

Description — Reads data records from an input pipe.

Syntax — Syntax varies with the programming language.

Examples — **COBOL II** `CALL 'GETPIPE' USING SPAREA.`

Usage — The GETPIPE command obtains the next result row and places the description in the SQLDA.

OPENPIPE

Description

Opens a data pipe to receive input from the remote DBMS through Open ClientConnect.

Syntax

Syntax varies with the programming language.

Parameters

The OPENPIPE command uses the values in these SPAREA fields:

- SPMODE (see page 88) specifies that the data pipe is opened for input.
- SPFORMAT (see page "SPFORMAT" on page 88) specifies that DB2 is the data pipe format.
- SPSQLDA (see page "SPSQLDA" on page 88) contains the address of a SQLDA that describes the content of the data records, being returned to the CSA.

Examples

COBOL II

```
MOVE 'INPUT'    TO SPMODE.
MOVE 'DB2'      TO SPFORMAT.
CALL 'OPENPIPE' USING SPAREA.
SET ADDRESS OF SQLDA TO SPSQLDA.
```

Usage

- CSA data pipes are always opened in input mode and DB2 format.
- If you transfer data between data sources with the INSERT, UPDATE, or TRANSFER command, you do not need to open a data pipe; you can simply send the Open ClientConnect TRANSFER command.
- Use the OPENPIPE command only when the CSA is expecting data results; in other words, after the CSA sends a SELECT request and results are available. To determine whether results are available, the CSA must check the SPSTATUS field (see page "SPSTATUS" on page 88) for 'R'.
- Open ClientConnect provides the SQLDA address. You define the *SQLDA* mask in the CSA code, and then pass it the address from Open ClientConnect.

REQEXEC

Description

Sends the contents of the request buffer (for example, a TRANSFER statement, a SQL statement, or any other valid request) to run against the target database.

Syntax

Syntax varies with the programming language.

Parameters

The REQEXEC command uses values from these SPAREA fields:

- *SPSQL* (see page "SPSQL" on page 89) specifies the address of a buffer containing one or more SQL statements.
- *SPSTATUS* (see page 88) specifies the status of the results returned.

Examples

COBOL II

```
SET ADDRESS OF SQL-BUFFER    TO SQL-RPT-PTR.
SET SPSQL                    TO SQL-RPT-PTR.
CALL 'REQEXEC' USING SPAREA.
```

Usage

- When the CSA issues the REQEXEC command, Open ClientConnect places result information in the *SPSTATUS* field. The CSA can examine this field to determine if more results are available. The possible *SPSTATUS* values are:
  - 'OK' indicates success.
  - 'E' indicates an error.
  - 'W' indicates a warning.
  - 'R' indicates a result is available.
  - [Blank] indicates a result is not available.
- The first 2 bytes (half-word) of the SQL request buffer must specify the total length of the buffer. If the buffer contains multiple SQL statements, the statements must be delimited by a semicolon (;).
- The maximum size for the SQL statements is 32KB.
- The CLIENTC2 sample program uses the REQEXEC command. The following example is excerpted from CLIENTC2's paragraph 55500-SEND-REQUEST.

RESCHECK

Description

Checks for multiple results from the execution of its SQL request(s).

Syntax

Syntax varies with the programming language.

Parameters

RESCHECK

The RESCHECK command uses the value from the SPAREA field SPSTATUS (see "SPSTATUS" on page 88), which specifies the status of the results returned.

Examples

COBOL II `CALL 'RESCHECK' USING SPAREA.`

Usage

- One RESCHECK must be issued for each expected result after the first result is received. If you expect the CSA to receive only one result, then you do not need to issue this command.

  Note The REQEXEC command provides the result information for the first or only SQL statement.

- When the CSA issues the RESCHECK command, Mainframe Client Connect places result information in the *SPSTATUS* field. The CSA can examine this field to determine if more results are available. The possible values are:
  - 'OK' indicates success.
  - 'E' indicates an error.
  - 'W' indicates a warning.
  - 'R' indicates a result is available.
  - [Blank] indicates no result is available.

APPENDIX B

CLIENTC2 Sample CSA

This appendix discusses the following topics:

- Using input pipes: about the CLIENTC2 sample code
- CLIENTC2 sample code

Using input pipes: about the CLIENTC2 sample code

The CLIENTC2 sample CSA issues a request and receives results from an Adaptive Server Enterprise database table. It illustrates:

- Specifying an attachment definition name
- Using an SPAREA definition
- Setting up a request buffer and executing requests
- Defining an input data pipe and SQLDA template and using a SQLDA definition provided by Open ClientConnect to receive results
- Using the RESCHECK and GETMSG commands to ensure the SQL requests are successful and that error handling works

CLIENTC2 uses a DB2 input pipe and a SQLDA definition to receive results from a remote server (in this case, Adaptive Server Enterprise). The SQLDA definition provides data structure information that is sent with the data to the CSA.

CLIENTC2 sample code

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.  CLIENTC2.
 ************************************************************
```

```
 * SAMPLE CSA PROGRAM TO ILLUSTRATE ESTABLISHING A
 * ConnectION TO A REMOTE SERVER, EXECUTING SQL REQUEST,
 * RETRIEVING THE RESULTS AND ANY MESSAGES (WRITING THEM TO A TEMP
 * STORAGE QUEUE STRICTLY FOR EXAMPLE), AND THEN DETACHING
 * FROM THE REMOTE SERVER.
 *
 ****************************************************************
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
****************************************************************
 * POINTERS.
 ****************************************************************
01 WS-POINTERS.
     05 SPAREA-PTR               POINTER.
     05 SQLDA-PTR                POINTER.
     05 SQL-REQ-PTR              POINTER.
****************************************************************
 * COUNTERS AND VARIOUS INTEGERS.
 ****************************************************************
01 WS-VARIABLES.
     05  CICSRC                  PIC S9(8) COMP VALUE +0.
     05  RESCHECK-CNT            PIC 9(03) VALUE ZEROES.
     05  SPSTATUS-CNT            PIC 9(03) VALUE ZEROES.
****************************************************************
 * ATTACHMENT DEFINITION NAME.  FOR SIMPLICITY OF EXAMPLE,
 * THIS PROGRAM ASSUMES THAT THE ATTACHMENT RECORD CONTAINS THE
 * USERID AND PASSWORD (OR ELSE THAT NONE ARE NEEDED).  IN AN
 * ACTUAL PRODUCTION ENVIRONMENT, USERID, PASSWORD, OR BOTH COULD
 * BE SPECIFIED AT RUNTIME.
 ****************************************************************
01 ATTACH-NAME                  PIC X(32) VALUE 'SQLSERVE'.
****************************************************************
 * SQL STATEMENT TO EXECUTE.  WILL SELECT ALL ROWS FROM THE SQL
 * SERVER SAMPLE PUBS DATABASE TABLE "SALES."  MULTIPLE SQL
 * REQUESTS CAN BE SENT IN ONE REQUEST BUFFER, AS LONG AS THEY ARE
 * SEPARATED BY SEMICOLONS.
 ****************************************************************
01 WS-SELECT-STMT.
     05 SELECT-STMT              PIC X(80)
             VALUE 'SELECT * FROM pubs2.sales'.
****************************************************************
 * FLAGS.
 ****************************************************************
01 WS-RESCHECK-DONE-SW          PIC X(01) VALUE 'N'.
     88 RESCHECK-DONE            VALUE 'Y'.
```

```
     88 RESCHECK-NOT-DONE        VALUE 'N'.
     88 LAST-SPSTATUS-SPACES     VALUE ' '.
01 WS-SPAREA-INIT-SW            PIC X(01) VALUE 'N'.
     88 SPAREA-INIT-OK           VALUE 'Y'.
     88 SPAREA-INIT-BAD          VALUE 'N'.
01 WS-ATTACH-TO-SERVER-SW       PIC X(01) VALUE 'N'.
     88 ATTACH-OK                VALUE 'Y'.
     88 ATTACH-FAILED            VALUE 'N'.
*****************************************************************
 * ERROR MESSAGES.
 *****************************************************************
01 WS-STUB-ERROR-MSG.
     03 FILLER                   PIC X(09)
            VALUE 'CALL TO: '
     03 WS-STUB-NAME             PIC X(08) VALUE SPACES.
     03 FILLER                   PIC X(18)
            VALUE ' - RECEIVED SPRC: '
     03 WS-STUB-SPRC             PIC X(03) VALUE '000'.
     03 FILLER                   PIC X(03) VALUE ' - '.
     03 WS-STUB-SPMSG            PIC X(100) VALUE SPACES.
01 WS-RESCHECK-MSG.
     03 FILLER                   PIC X(33)
            VALUE 'RESCHECK NON-BLANK STATUS - REC: '.
     03 WS-RESCHECK-NUMBER       PIC 9(03) VALUE ZEROES.
     03 FILLER                   PIC X(16)
            VALUE ' - SPSTATUS IS: '.
     03 WS-SPSTATUS-OUT          PIC X(02) VALUE SPACES.
     03 FILLER                   PIC X(11)
            VALUE ' - SPCODE: '.
     03 WS-SPCODE-OUT            PIC X(03) VALUE SPACES.
     03 FILLER                   PIC X(10)
            VALUE ' - SPIND: '.
     03 WS-SPIND-OUT             PIC X(01) VALUE SPACE.
*****************************************************************
 * FORMATTED SALES DATA RECORD TO BE WRITTEN TO TEMP STORAGE.
 *****************************************************************
01 SALES-ROW.
     03 SALES-STOR-ID            PIC X(04) VALUE SPACES.
     03 FILLER                   PIC X(01) VALUE SPACE.
     03 SALES-ORD-NUM            PIC X(20) VALUE SPACES.
     03 FILLER                   PIC X(01) VALUE SPACE.
     03 SALES-DATE               PIC X(10) VALUE SPACES.
     03 FILLER                   PIC X(01) VALUE SPACE.
     03 SALES-QTY                PIC 9(04) VALUE 0.
     03 FILLER                   PIC X(01) VALUE SPACE.
     03 SALES-PAY-TERMS          PIC X(12) VALUE SPACES.
```

```
     03 FILLER                    PIC X(01) VALUE SPACE.
     03 SALES-TITLE-ID            PIC X(06) VALUE SPACES.
LINKAGE SECTION.
*****************************************************************
 * UNLIKE A NORMAL RSP, WHERE THE SPAREA IS SUPPLIED VIA THE COMM
 * AREA BY MAINFRAMEConnect BEFORE LINKING TO THE RSP, IN A CSA,
 * THIS PROGRAM SUPPLIES THE SPAREA, WHICH IS
 * THEN INITIALIZED BY THE CALL TO CSSETUP.
 *****************************************************************
01 STORE-PROC-AREA.
     COPY SPAREAC.
*****************************************************************
 * SQL REQUEST BUFFER THAT WILL BE PASSED TO THE REMOTE SERVER VIA
 * REQEXEC CALL.  IT CONSISTS OF A HALFWORD LENGTH FIELD, AND THE
 * ACTUAL REQUEST STATEMENT.
 *****************************************************************
01 SQL-BUFFER.
     03 SQL-LENGTH                PIC S9(4) COMP.
     03 SQL-REQUEST               PIC X(80).
*****************************************************************
 * SQLDA FOR DB2-FORMAT INPUT PIPE THAT WILL RETURN THE RESULT
 * ROWS FROM THE SALES TABLE.  HARDCODED FOR SIX OCCURENCES OF
 * SQLVAR SINCE WE KNOW AHEAD OF TIME THAT IS THE NUMBER OF
 * COLUMNS THE SALES TABLE HAS.  THE ACTUAL SQLDA WILL BE BUILT
 * AND A POINTER SUPPLIED TO IT WHEN WE OPEN THE DB2-FORMAT INPUT
 * PIPE TO READ RESULTS.
 *****************************************************************
01 SALES-SQLDA.
     03 SALES-SQLDAID             PIC X(08).
     03 SALES-SQLDABC             PIC S9(8) COMP.
     03 SALES-SQLN                PIC S9(4) COMP.
     03 SALES-SQLD                PIC S9(4) COMP.
     03 SALES-SQLVAR              OCCURS 6 TIMES.
         05 SALES-SQLTYPE         PIC S9(4) COMP.
         05 SALES-SQLLEN          PIC S9(4) COMP.
         05 SALES-SQLDATA         POINTER.
         05 SALES-SQLIND          POINTER.
         05 SALES-SQLNAME         PIC X(32).
*****************************************************************
 * DATA FIELDS POINTED TO BY THE SQLDATA POINTERS.
 * NOTE THAT VARCHAR FIELDS ARE PRECEEDED BY A LENGTH FIELD.
 * OTHER DATATYPES HAVE THEIR OWN REQUIREMENTS. CHECK THE IBM
 * DXT REFERENCE MANUAL OR MODELRSP IN THE RSP PROGRAMMER'S REF.
 *****************************************************************
01 STORE-ID                     PIC X(04).
01 ORDER-NUMBER.
```

```
     03 ORD-NUM-LENGTH           PIC S9(4) COMP.
     03 ORD-NUM.
         05 ORD-NUMCHAR          PIC X(01)
            OCCURS 20 TIMES DEPENDING ON ORD-NUM-LENGTH.
01 ORDER-DATE                   PIC X(10).
01 QUANTITY                     PIC S9(4) COMP.
01  PAY-TERMS.
     03 PAY-TERM-LEN             PIC S9(4) COMP.
     03 PAY-TERM.
        05 PAY-TERM-CHAR         PIC X(01)
             OCCURS 12 TIMES DEPENDING ON PAY-TERM-LEN.
01  TITLE-ID-ENT.
     03 TITLE-ID-LEN             PIC S9(4) COMP.
     03 TITLE-ID.
         05 TITLE-ID-CHAR        PIC X(01)
             OCCURS 6 TIMES DEPENDING ON TITLE-ID-LEN.
*****************************************************************
 PROCEDURE DIVISION.
 *****************************************************************
0000-MAIN-PROCESSING.
    PERFORM 1000-SPAREA-INIT        THRU 1000-EXIT.
    PERFORM 5000-PROCESS-REQUEST    THRU 5000-EXIT.
0000-GET-OUT-NOW.
    EXEC CICS
         RETURN
     END-EXEC.
0000-EXIT.
     EXIT.
*****************************************************************
 * GET AN SPAREA, AND CALL CSA TO INITIALIZE IT.
 *****************************************************************
1000-SPAREA-INIT.
    EXEC CICS
         DELETEQ TS QUEUE('CSEXQUE')
         RESP(CICSRC)
     END-EXEC.
*****************************************************************
 * THIS GETMAIN MAKES THE SPAREA AVAILABLE TO OPEN CLIENTConnect.
 *****************************************************************
    EXEC CICS
         GETMAIN SET(SPAREA-PTR)
         LENGTH(LENGTH OF SPAREA) NOSUSPEND
         RESP(CICSRC)
     END-EXEC.
    IF CICSRC = DFHRESP(NORMAL)
         SET ADDRESS OF STORE-PROC-AREA TO SPAREA-PTR
```

```
         PERFORM 1100-CALL-CSSETUP      THRU 1100-EXIT
     END-IF.
1000-EXIT.
     EXIT.
*****************************************************************
 * CALL CSA TO INITIALIZE SPAREA
 *****************************************************************
1100-CALL-CSSETUP.
    MOVE 'Y' TO SPTRCOPT.
    CALL 'CSSETUP' USING SPAREA.
    IF SPRC = '000'
         MOVE 'Y'                       TO WS-SPAREA-INIT-SW
     ELSE
         MOVE 'CSSETUP'                 TO WS-STUB-NAME
         PERFORM 6900-STUB-CALL-ERROR   THRU 6900-EXIT
         GO TO 0000-GET-OUT-NOW
     END-IF.


1100-EXIT.
     EXIT.
*****************************************************************
 * CONTROL THE PROCESS OF ATTACHING TO SQL SERVER, EXECUTING THE
 * SELECT REQUEST, AND RETRIEVING THE RESULTS.
 *****************************************************************
5000-PROCESS-REQUEST.
    PERFORM 5100-ATTACH-TO-SQL-SERVER  THRU 5100-EXIT.
    IF ATTACH-OK
         PERFORM 5500-SEND-REQUEST      THRU 5500-EXIT
         PERFORM 5700-READ-RESULTS      THRU 5700-EXIT
         PERFORM 5800-CALL-DETACH       THRU 5800-EXIT.
5000-EXIT.
     EXIT.
*****************************************************************
 * CALL CLIENT SERVICES TO ATTACH TO THE SQL SERVER.
 *****************************************************************
5100-ATTACH-TO-SQL-SERVER.
MOVE ATTACH-NAME                    TO SPSERVER.
     CALL 'ATTACH' USING SPAREA.
    IF SPRC = '000'
         MOVE 'Y'                       TO WS-ATTACH-TO-SERVER-SW
     ELSE
         MOVE 'ATTACH'                  TO WS-STUB-NAME
         PERFORM 6900-STUB-CALL-ERROR   THRU 6900-EXIT
         GO TO 0000-GET-OUT-NOW
     END-IF
```

```
5100-EXIT.
     EXIT.
*****************************************************************
 * EXECUTE THE SQL REQUEST AGAINST THE REMOTE SERVER.
 *****************************************************************
5500-SEND-REQUEST.


    EXEC CICS
         GETMAIN SET(SQL-REQ-PTR)
         LENGTH(LENGTH OF SQL-BUFFER)
         NOSUSPEND RESP(CICSRC)
     END-EXEC.
    SET ADDRESS OF SQL-BUFFER         TO SQL-REQ-PTR.
     SET SPSQL                         TO SQL-REQ-PTR.
    IF CICSRC = DFHRESP(NORMAL)
         MOVE SELECT-STMT              TO SQL-REQUEST
         MOVE LENGTH OF SELECT-STMT    TO SQL-LENGTH
         CALL 'REQEXEC' USING SPAREA
         IF SPRC = '000'
             PERFORM 6000-RESCHECK-SEARCH THRU 6000-EXIT
                 UNTIL RESCHECK-DONE
         END-IF
     END-IF.
5500-EXIT.
     EXIT.
*****************************************************************
 * RETRIEVE ANY RESULT ROWS BY OPENING DB2 INPUT PIPE.
 *****************************************************************
5700-READ-RESULTS.
    PERFORM 5710-OPEN-GETPIPE         THRU 5710-EXIT.
     PERFORM 5720-GETPIPE-LOOP         THRU 5720-EXIT
         UNTIL SPRC NOT = '000'.
5700-EXIT.
     EXIT.
*****************************************************************
 * OPEN THE DB2 INPUT PIPE.
 *****************************************************************
5710-OPEN-GETPIPE.
    MOVE 'INPUT '                     TO SPMODE.
     MOVE 'DB2'                        TO SPFORMAT.
     CALL 'OPENPIPE' USING SPAREA.
    IF SPRC = '000'
         SET ADDRESS OF SALES-SQLDA    TO SPSQLDA
        SET ADDRESS OF STORE-ID       TO SALES-SQLDATA(1)
         SET ADDRESS OF ORDER-NUMBER   TO SALES-SQLDATA(2)
```

```
         SET ADDRESS OF ORDER-DATE     TO SALES-SQLDATA(3)
         SET ADDRESS OF QUANTITY       TO SALES-SQLDATA(4)
         SET ADDRESS OF PAY-TERMS      TO SALES-SQLDATA(5)
         SET ADDRESS OF TITLE-ID-ENT   TO SALES-SQLDATA(6)
     ELSE
         MOVE 'OPENPIPE'               TO WS-STUB-NAME
         PERFORM 6900-STUB-CALL-ERROR  THRU 6900-EXIT
         GO TO 0000-GET-OUT-NOW
     END-IF.
5710-EXIT.
     EXIT.
*****************************************************************
 USE GETPIPE TO RETRIEVE ANY RESULT ROWS.
 ****************************************************************5720-
GETPIPE-LOOP.
    CALL 'GETPIPE' USING SPAREA.
    IF SPRC = '000'
         MOVE SPACES             TO SALES-ROW
         MOVE STORE-ID                 TO SALES-STOR-ID
         MOVE ORD-NUM                  TO SALES-ORD-NUM
         MOVE ORDER-DATE               TO SALES-DATE
         MOVE QUANTITY                 TO SALES-QTY
         MOVE PAY-TERM                 TO SALES-PAY-TERMS
         MOVE TITLE-ID                 TO SALES-TITLE-ID
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE')
                 FROM(SALES-ROW) NOSUSPEND
             RESP(CICSRC)
         END-EXEC
    ELSE
         IF SPRC NOT = 'EOF'
             MOVE 'GETPIPE'            TO WS-STUB-NAME
             PERFORM 6900-STUB-CALL-ERROR THRU 6900-EXIT
             GO TO 0000-GET-OUT-NOW
         END-IF
     END-IF.
    IF SPIND = 'M'
         PERFORM 6100-GET-MESSAGES     THRU 6100-EXIT
             UNTIL SPIND NOT = 'M'
     END-IF.
5700-EXIT.
     EXIT.
*****************************************************************
 * CALL THE DETACH STUB TO DETACH FROM A REMOTE SERVER.
 ****************************************************************
5800-CALL-DETACH.
```

```
    CALL 'DETACH' USING SPAREA.
    IF SPRC NOT = '000'
         MOVE 'DETACH'                  TO WS-STUB-NAME
         PERFORM 6900-STUB-CALL-ERROR  THRU 6900-EXIT
         GO TO 0000-GET-OUT-NOW.
5800-EXIT.
     EXIT.
*****************************************************************
 * CHECK RESCHECK TO LOCATE ANY ERROR MESSAGES RETURNED FROM LAN.
 * ALWAYS LOG ANY NON-ZERO STATUS AFTER A CALL TO REQEXEC.
 *****************************************************************
6000-RESCHECK-SEARCH.
    ADD 1                              TO RESCHECK-CNT.
    IF SPSTATUS NOT = ' '(space)
          MOVE 'N'                      TO WS-RESCHECK-DONE-SW
         ADD 1                         TO SPSTATUS-CNT
         MOVE SPSTATUS-CNT             TO WS-RESCHECK-NUMBER
         MOVE SPSTATUS                 TO WS-SPSTATUS-OUT
         MOVE SPCODE                   TO WS-SPCODE-OUT
         MOVE SPIND                    TO WS-SPIND-OUT
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE')
                 FROM(WS-RESCHECK-MSG) NOSUSPEND
                 RESP(CICSRC)
         END-EXEC
         IF SPIND NOT = SPACES
             PERFORM 6100-GET-MESSAGES THRU 6100-EXIT
                 UNTIL SPIND NOT = 'M'
         END-IF
    ELSE
         IF LAST-SPSTATUS-SPACES
             MOVE 'Y'                  TO WS-RESCHECK-DONE-SW
         ELSE
             MOVE ' '                  TO WS-RESCHECK-DONE-SW
         END-IF
     END-IF.
    IF NOT RESCHECK-DONE
         CALL 'RESCHECK' USING SPAREA.
6000-EXIT.
     EXIT.
*****************************************************************
 * RETRIEVE ANY OUTSTANDING MESSAGES FOR A REQUEST.
 *****************************************************************
6100-GET-MESSAGES.
        CALL 'GETMSG' USING SPAREA
    IF SPMSG NOT = SPACES
```

```
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE')
                 FROM(SPMSG) NOSUSPEND
             RESP(CICSRC)
         END-EXEC
     END-IF.
6100-EXIT.
     EXIT.
*****************************************************************
 * FORMAT AND WRITE STUB-CALL ERROR INFO TO TS QUEUE.
 *****************************************************************
6900-STUB-CALL-ERROR.
    MOVE SPRC                          TO WS-STUB-SPRC.
     MOVE SPMSG                         TO WS-STUB-SPMSG.
    EXEC CICS
         WRITEQ TS QUEUE('CSEXQUE')
             FROM(WS-STUB-ERROR-MSG) NOSUSPEND
         RESP(CICSRC)
     END-EXEC.
6900-EXIT.
     EXIT.
```

APPENDIX C

CSAINDX Sample CSA

This appendix discusses the following topics:

- Transferring data: about the CSAINDX sample code
- CSAINDX sample code
- Detailed explanation of the sample code

Transferring data: about the CSAINDX sample code

The CSAINDX sample CSA transfers data retrieved from an RSP executed through DirectConnect to Adaptive Server Enterprise. Transferring data between data sources is one of the most common uses of CSAs.

This program illustrates:

- Specifying two attachment definition names, one for DirectConnect and one for Adaptive Server Enterprise
- Using an SPAREA definition
- Setting up a request buffer and executing requests
- Using the RESCHECK and GETMSG commands to ensure that the DirectConnect requests are successful handling errors

If you transfer data between sources regularly, you can set up a job scheduler to invoke the CSA on the mainframe.

CSAINDX sample code

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  CSAINDX.
*****************************************************************
```

```
 * SAMPLE CSA PROGRAM TO ILLUSTRATE:
 * 1) Connect TO THE DIRECTConnect, JUST TO VERIFY IT'S UP
 * 2) Connect TO SQL SERVER, EXECUTING A STORED PROCEDURE
 *    THAT DELETES AN INDEX ON A SQL SERVER TABLE
 * 3) Connect TO THE DIRECTConnect, DO A TRANSFER TO THAT TABLE
 * 4) Connect TO SQL SERVER, EXECUTING A STORED PROCEDURE
 *    THAT RE-CREATES AN INDEX ON THAT SQL SERVER TABLE
 *
 * TRANSID IN PCT: PIDX     PROGRAM NAME IN PPT: CSAINDX
 ****************************************************************
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
****************************************************************
 * POINTERS.
 ****************************************************************
01 WS-POINTERS.
     05 SPAREA-PTR                  POINTER.
     05 SQLDA-PTR                   POINTER.
     05 SQL-REQ-PTR                 POINTER.
****************************************************************
 * COUNTERS AND VARIOUS INTEGERS.
 ****************************************************************
01 WS-VARIABLES.
     05 CICSRC                      PIC S9(8) COMP VALUE +0.
     05 RESCHECK-CNT                PIC 9(3) VALUE ZEROES.
     05 SPSTATUS-CNT                PIC 9(3) VALUE ZEROES.
****************************************************************
 * ATTACHMENT DEFINITION NAME.
 ****************************************************************
01 WS-ATTACH-NAMES.
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *
 * ATTNAME-1 - AN ATTACHMENT RECORD WITH THE DIRECTConnect SERVICENAME *
 * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *
    05 ATTNAME-1                   PIC X(08) VALUE 'GWSERVNM'.
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *
 * ATTNAME-2 - ATTACHMENT RECORD WITH THE SQL SERVER SERVICENAME *
 * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *
    05 ATTNAME-2                   PIC X(08) VALUE 'SQLSERVE'.
****************************************************************
 * SWITCH FOR RESCHECK READS -
 *     THE IDEA IS TO KEEP CALLING RESCHECK UNTIL YOU'VE RECEIVED
 *     SPACES IN SPSTATUS TWICE IN A ROW - THEN YOU'RE DONE.
 *
 *     IMPORTANT FOR "BATCH" COMMANDS SENT THRU THE SQL BUFFER.
 ****************************************************************
```

```
01 WS-SWITCHES.
         03 WS-RESCHECK-DONE-SW          PIC X VALUE 'N'.
             88 RESCHECK-DONE            VALUE 'Y'.
             88 RESCHECK-NOT-DONE        VALUE 'N'.
             88 LAST-SPSTATUS-SPACES     VALUE ' '.
        03 WS-INIT-OKAY-SW              PIC X(01) VALUE 'Y'.
             88 SPAREA-INIT-OK           VALUE 'Y'.
             88 SPAREA-INIT-BAD          VALUE 'N'.
        03 WS-ATTACH-OK-SW              PIC X(01) VALUE 'Y'.
             88 ATTACH-OKAY              VALUE 'Y'.
             88 ATTACH-FAILED            VALUE 'N'.
*****************************************************************
 * SQL STATEMENT TO EXECUTE. - ALL STATEMENTS ARE 90 BYTES.
 * BEAR IN MIND COMMANDS SENT TO SYBASE SQL SERVER MAY BE CASE SENSITIVE.
 *****************************************************************
01 SQL-BUFFER-CMDS.
         03 DELETE-IDX-SP-STMT.
             05 FILLER      PIC X(16) VALUE
             'execute rickdinx'.
             05 FILLER      PIC X(74) VALUE SPACES.
         03 CREATE-IDX-SP-STMT.
             05 FILLER      PIC X(16) VALUE
             'execute rickcinx'.
             05 FILLER      PIC X(74) VALUE SPACES.
         03 TRANSFER-STMT.
             05 FILLER      PIC X(12) VALUE
             'TRANSFER TO '.
             05 FILLER      PIC X(01) VALUE QUOTE.
             05 FILLER      PIC X(15) VALUE
             'trex ssuid sspw'.
             05 FILLER      PIC X(01) VALUE QUOTE.
             05 FILLER      PIC X(02) VALUE '; '.
             05 FILLER      PIC X(26) VALUE
             'WITH REPLACE INTO samp04in'.
             05 FILLER      PIC X(02) VALUE '; '.
             05 FILLER      PIC X(30) VALUE
             'USE PROCEDURE SAMP04C 00200   '.
             05 FILLER      PIC X(01) VALUE SPACES.


*****************************************************************
 * ERROR MESSAGES.
 *****************************************************************
01 WS-ATTACH-ERR-MSG.
         03 FILLER               PIC X(15)
           VALUE 'ATTACHMENT TO: '.
```

```
         03 WS-ATTACH-ERR-NAME   PIC X(08) VALUE SPACES.
         03 FILLER               PIC X(20)
           VALUE ' - WITH ERROR CODE: '.
         03 WS-ATTACH-ERR-MSGCODE PIC X(03) VALUE '000'.
01 WS-RUN-COUNT-MSG.
         03 FILLER               PIC X(30)
           VALUE '* STARTING TRANSFER LOOP NUM: '.
         03 WS-RUN-COUNT         PIC 9(03) VALUE ZEROES.
01 WS-CSSETUP-ERROR-MSG.
         03 FILLER               PIC X(30)
           VALUE '! CSSETUP HAD A PROBLEM SPRC: '.
         03 WS-CSSETUP-SPRC      PIC X(03) VALUE SPACES.
01 WS-RESCHECK-LAST-MSG.
         03 FILLER               PIC X(30)
           VALUE '- FINAL RESCHECK READ COUNT : '.
         03 WS-RESCHECK-COUNT    PIC 9(03) VALUE ZEROES.
01 WS-DID-SYBASE-DELETE.
         03 FILLER               PIC X(30)
           VALUE '> DID SYBASE DELETE INDEXES SP'.
01 WS-DID-GW-TRANSFER.
         03 FILLER               PIC X(30)
           VALUE '> DID GW TRANSFER RSP TO SYBAS'.
01 WS-DID-SYBASE-CREATE.
         03 FILLER               PIC X(30)
           VALUE '> DID SYBASE CREATE INDEXES SP'.
01 WS-RESCHECK-MSG.
         03 FILLER               PIC X(33)
           VALUE 'RESCHECK NON-BLANK STATUS - REC: '.
         03 WS-RESCHECK-NUMBERPIC 9(03) VALUE ZEROES.
         03 FILLER               PIC X(16)
           VALUE ' - SPSTATUS IS: '.
         03 WS-SPSTATUS-OUTPIC X(02) VALUE SPACES.
         03 FILLER               PIC X(11)
           VALUE  ' - SPCODE: '.
         03 WS-SPCODE-OUT        PIC X(03) VALUE SPACES.
         03 FILLER               PIC X(10)
           VALUE ' - SPIND: '.
         03 WS-SPIND-OUT         PIC X(01) VALUE SPACE.


LINKAGE SECTION.


01 STORED-PROC-AREA.
    COPY SPAREAC.
*****************************************************************
```

```
 * SQL REQUEST BUFFER THAT WILL BE PASSED TO THE REMOTE SERVER VIA
 * REQEXEC CALL.  IT CONSISTS OF A HALFWORD LENGTH FIELD, AND THE
 * ACTUAL REQUEST STATEMENT.
 ****************************************************************
01  SQL-BUFFER.
        03 SQL-LENGTH           PIC S9(4) COMP.
         03 SQL-REQUEST          PIC X(100).
*================================================================*
 PROCEDURE DIVISION.
 *================================================================*
0000-MAIN-PROCESSING.
    PERFORM 1000-SPAREA-INIT.
    IF ATTACH-OKAY
         PERFORM 5000-TRANSFER-PROCESS THRU 5000-EXIT
         PERFORM 9900-FINALCOUNT       THRU 9900-EXIT
     END-IF.
0000-GET-OUT-NOW.
    EXEC CICS
         RETURN
     END-EXEC.
0000-EXIT.
     EXIT.
****************************************************************
 * GET AN SPAREA, AND CALL CSA TO INITIALIZE IT.
 ****************************************************************
1000-SPAREA-INIT.
    EXEC CICS
         DELETEQ TS QUEUE('CSEXQUE') RESP(CICSRC)
     END-EXEC.
    EXEC CICS
         GETMAIN SET(SPAREA-PTR)
         LENGTH(LENGTH OF SPAREA)
         NOSUSPEND
         RESP(CICSRC)
     END-EXEC.
    IF CICSRC = DFHRESP(NORMAL)
         SET ADDRESS OF STORED-PROC-AREA TO SPAREA-PTR
         PERFORM 1100-CALL-CSSETUP   THRU 1100-EXIT
     END-IF.
    EXEC CICS
         GETMAIN SET(SQL-REQ-PTR)
         LENGTH(LENGTH OF SQL-BUFFER)
         NOSUSPEND RESP(CICSRC)
     END-EXEC.
    IF CICSRC = DFHRESP(NORMAL)
         SET ADDRESS OF SQL-BUFFER TO SQL-REQ-PTR
```

```
         SET SPSQL TO SQL-REQ-PTR
     ELSE
         MOVE 'N'       TO WS-INIT-OKAY-SW
     END-IF.
1000-EXIT.
     EXIT.
*****************************************************************
 * CALL CSA TO INITIALIZE SPAREA
 *****************************************************************
1100-CALL-CSSETUP.
    CALL 'CSSETUP' USING SPAREA.
    IF SPRC = '000'
         MOVE 'Y'      TO WS-INIT-OKAY-SW
     ELSE
         MOVE SPRC     TO WS-CSSETUP-SPRC
     EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE')
                 FROM(WS-CSSETUP-ERROR-MSG) NOSUSPEND
                 RESP(CICSRC)
         END-EXEC
         GO TO 0000-GET-OUT-NOW
     END-IF.
1100-EXIT.
     EXIT.
*****************************************************************
 * CONTROL THE PROCESS OF ATTACH, EXEC, DETATCH FOR TRANSFER.
 * 1) 1ST ATTACH TO DIRECTConnect CHECKS IF IT'S ALIVE AND WELL.
 * 2) 2ND ATTACH TO SQL SERVER - RUNS S.P. TO DELETE INDEXES.
 * 3) 3RD ATTACH TO DIRECTConnect RUNS TRANSFER FROM RSP TO SYBASE SQL SERVER.
 * 4) 4TH ATTACH TO SQL SERVER - RUNS S.P. TO RE-CREATE INDEXES.
 *****************************************************************
5000-TRANSFER-PROCESS.
    PERFORM 5100-WRITE-RUN-COUNT          THRU 5100-EXIT.
    PERFORM 5600-ATTACH-TO-DIRECTConnect THRU 5600-EXIT.
     PERFORM 5800-CALL-DETACH              THRU 5800-EXIT.
    PERFORM 5700-ATTACH-TO-SYBASE SQL SERVER THRU 5700-EXIT.
     PERFORM 5200-LOAD-DEL-INDEX-STMT      THRU 5200-EXIT.
     PERFORM 5500-CALL-REQEXEC             THRU 5500-EXIT.
     PERFORM 5800-CALL-DETACH              THRU 5800-EXIT.
    PERFORM 5600-ATTACH-TO-DIRECTConnect THRU 5600-EXIT.
     PERFORM 5300-LOAD-TRANSFER-STMT       THRU 5300-EXIT.
     PERFORM 5500-CALL-REQEXEC             THRU 5500-EXIT.
     PERFORM 5800-CALL-DETACH              THRU 5800-EXIT.
    PERFORM 5700-ATTACH-TO-SYBASE         THRU 5700-EXIT.
     PERFORM 5400-LOAD-CRE-INDEX-STMT      THRU 5400-EXIT.
     PERFORM 5500-CALL-REQEXEC             THRU 5500-EXIT.
```

```
     PERFORM 5800-CALL-DETACH              THRU 5800-EXIT.
5000-EXIT.
     EXIT.
*****************************************************************
 * SEND THE TRANSFER LOOP RUN COUNT TO TEMP STORAGE QUE
 *****************************************************************
5100-WRITE-RUN-COUNT.
    ADD 1               TO WS-RUN-COUNT.
    EXEC CICS
         WRITEQ TS QUEUE('CSEXQUE')
             FROM(WS-RUN-COUNT-MSG) NOSUSPEND
             RESP(CICSRC)
     END-EXEC.
5100-EXIT.
     EXIT.
*****************************************************************
 * LOAD THE DELETE INDEX STATEMENT INTO THE SQL BUFFER
 *****************************************************************
5200-LOAD-DEL-INDEX-STMT.
    MOVE DELETE-IDX-SP-STMT              TO SQL-REQUEST.
     MOVE LENGTH OF DELETE-IDX-SP-STMT    TO SQL-LENGTH.
     PERFORM 5500-CALL-REQEXEC            THRU 5500-EXIT.
5200-EXIT.
     EXIT.
*****************************************************************
 * LOAD THE TRANSFER STATEMENT INTO THE SQL BUFFER FOR THE DIRECTConnect
 *****************************************************************
5300-LOAD-TRANSFER-STMT.
    MOVE TRANSFER-STMT                   TO SQL-REQUEST.
     MOVE LENGTH OF TRANSFER-STMT         TO SQL-LENGTH.
     PERFORM 5500-CALL-REQEXEC            THRU 5500-EXIT.
5300-EXIT.
     EXIT.
*****************************************************************
 * LOAD THE CREATE INDEX STATEMENT INTO THE SQL BUFFER
 *****************************************************************
5400-LOAD-CRE-INDEX-STMT.
    MOVE CREATE-IDX-SP-STMT              TO SQL-REQUEST.
     MOVE LENGTH OF CREATE-IDX-SP-STMT    TO SQL-LENGTH.
     PERFORM 5500-CALL-REQEXEC            THRU 5500-EXIT.
5400-EXIT.
     EXIT.
*****************************************************************
 * EXECUTE THE SQL REQUEST AGAINST THE REMOTE SERVER.
 *****************************************************************
5500-CALL-REQEXEC.
```

```
    CALL 'REQEXEC' USING SPAREA.
    IF SPRC = '000'
         PERFORM 5900-RESCHECK-SEARCH     THRU 5900-EXIT
             UNTIL RESCHECK-DONE.
5500-EXIT.
     EXIT.
*****************************************************************
 * CALL CSA TO ATTACH TO DIRECTConnect.
 *****************************************************************
5600-ATTACH-TO-DIRECTConnect.
    MOVE ATTNAME-1         TO SPATTACH.
    CALL 'ATTACH' USING SPAREA.
    IF SPRC = '000'
         MOVE 'Y'           TO WS-ATTACH-OK-SW
     ELSE
         MOVE ATTNAME-1     TO WS-ATTACH-ERR-NAME
     MOVE SPRC              TO WS-ATTACH-ERR-MSGCODE
         EXEC CICS
             SEND FROM(WS-ATTACH-ERR-MSG) ERASE RESP(CICSRC)
         END-EXEC
         IF SPIND = 'M'
             PERFORM 9700-GET-MESSAGES
                 UNTIL SPIND NOT = 'M'
         END-IF
         GO TO 0000-GET-OUT-NOW
     END-IF.
5600-EXIT.
     EXIT.
*****************************************************************
 * CALL CSA TO ATTACH TO SYBASE SQL SERVER.
 *****************************************************************
5700-ATTACH-TO-SYBASE.
    MOVE ATTNAME-2         TO SPATTACH.
    CALL 'ATTACH' USING SPAREA.
    IF SPRC = '000'
     MOVE 'Y'               TO WS-ATTACH-OK-SW
     ELSE
         MOVE ATTNAME-2     TO WS-ATTACH-ERR-NAME
         MOVE SPRC          TO WS-ATTACH-ERR-MSGCODE
         EXEC CICS
             SEND FROM(WS-ATTACH-ERR-MSG) ERASE RESP(CICSRC)
         END-EXEC
         IF SPIND = 'M'
             PERFORM 9700-GET-MESSAGES
                 UNTIL SPIND NOT = 'M'
         END-IF
```

```
         GO TO 0000-GET-OUT-NOW
     END-IF.
5700-EXIT.
     EXIT.
*****************************************************************
 * CALL THE DETACH STUB TO DETACH FROM A REMOTE SERVER.
 *****************************************************************
5800-CALL-DETACH.
    CALL 'DETACH' USING SPAREA.
    IF SPRC NOT = '000'
         PERFORM 5900-RESCHECK-SEARCH      THRU 5900-EXIT
             UNTIL RESCHECK-DONE.
5800-EXIT.
     EXIT.
*****************************************************************
 * CHECK RESCHECK TO LOCATE SQL STATEMENT IN ERROR.
 * ALWAYS LOG ANY NON-ZERO STATUS AFTER ANY CALL TO SYBASE ICD STUBS.
 *****************************************************************
5900-RESCHECK-SEARCH.
    ADD 1                                TO RESCHECK-CNT.
    IF SPSTATUS NOT = '  '
         MOVE 'N'                         TO WS-RESCHECK-DONE-SW
         ADD 1                            TO SPSTATUS-CNT
         MOVE SPSTATUS-CNT                TO WS-RESCHECK-NUMBER
         MOVE SPSTATUS                    TO WS-SPSTATUS-OUT
         MOVE SPCODE                      TO WS-SPCODE-OUT
         MOVE SPIND                       TO WS-SPIND-OUT
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE')
                 FROM(WS-RESCHECK-MSG) NOSUSPEND
                 RESP(CICSRC)
         END-EXEC
         IF SPIND NOT = SPACES
             PERFORM 9700-GET-MESSAGES UNTIL SPIND NOT = 'M'
         END-IF
     ELSE
         IF LAST-SPSTATUS-SPACES
             MOVE 'Y'                     TO WS-RESCHECK-DONE-SW
         ELSE
             MOVE ' '                     TO WS-RESCHECK-DONE-SW
         END-IF
     END-IF.
        IF NOT RESCHECK-DONE
         CALL 'RESCHECK' USING SPAREA.
```

```
5900-EXIT.
     EXIT.
*****************************************************************
 * RETRIEVE ANY OUTSTANDING MESSAGES FOR A REQUEST.
 * LOG ALL MESSAGES TO TEMP STORAGE QUEUE FOR LATER EXAM - ALWAYS!
 *****************************************************************
9700-GET-MESSAGES.
    CALL 'GETMSG' USING SPAREA
     IF SPMSG NOT = SPACES
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE')
                 FROM(SPMSG) NOSUSPEND
             RESP(CICSRC)
         END-EXEC
     END-IF.
9700-EXIT.
     EXIT.
*****************************************************************
 * SEND THE FINAL RESCHECK READ NUMBER TO TEMP STORAGE QUEUE
 *****************************************************************
9900-FINALCOUNT.
    MOVE RESCHECK-CNT                    TO WS-RESCHECK-COUNT.
    EXEC CICS
        WRITEQ TS QUEUE('CSEXQUE')
             FROM(WS-RESCHECK-LAST-MSG) NOSUSPEND
             RESP(CICSRC)
     END-EXEC.
9900-EXIT.
     EXIT.
*===============================================================*
 *    END OF PROGRAM.
 *===============================================================*
```

Detailed explanation of the sample code

This section provides the following detail about the sample code:

- Using attachment definitions
- Transferring data
- Specifying error handling

Using attachment definitions

CSAINDX contains two attachment definitions in the WORKING-STORAGE SECTION: one for connecting to DB2 with DirectConnect and one for connecting to Adaptive Server Enterprise. If you want to write a CSA to transfer data between two LAN services, such as a DirectConnect and Adaptive Server Enterprise, you need two attachment definitions, as this CSAINDX example illustrates:

```
05  ATTNAME-1       PIC X(08) VALUE 'GWSERVNM'.
05  ATTNAME-2       PIC X(08) VALUE 'SQLSERVE'.
```

Transferring data

CSAINDX demonstrates how to transfer data between DB2 and Adaptive Server Enterprise. In this case, DB2, the primary database is the source, and Adaptive Server Enterprise, the secondary database, is the target. Therefore, the TRANSFER statement requires the format:

```
TRANSFER TO 'SQLServername user password'
WITH REPLACE INTO 'SQLServer_tablename'
@SQL REQUEST
```

In the sample program, the SQL request is actually an RSP called SAMP04C, which contains several SQL requests. CSAINDX codes the TRANSFER in the WORKING-STORAGE SECTION within 01 SQL-BUFFER-CMDS:

```
03 TRANSFER-STMT.
     05 FILLER             PIC X(12) VALUE
     'TRANSFER TO '.
     05 FILLER             PIC X(01) VALUE QUOTE.
     05 FILLER             PIC X(15) VALUE
     'trex ssuid sspw'.
     05 FILLER             PIC X(01) VALUE QUOTE.
     05 FILLER             PIC X(02) VALUE '; '.
     05 FILLER             PIC X(26) VALUE
     'WITH REPLACE INTO samp04in'.
     05 FILLER             PIC X(02) VALUE '; '.
     05 FILLER             PIC X(30) VALUE
     'USE PROCEDURE SAMP04C 00200 '.
     05 FILLER             PIC X(01) VALUE SPACES.
```

Note Your SQL statements cannot exceed the length you assign SQL-REQUEST in the LINKAGE SECTION.

You can also transfer data in the other direction—from Adaptive Server Enterprise to a supported database through DirectConnect. For details on the syntax and use of the TRANSFER statement, see the DirectConnect documentation.

After preparing the TRANSFER statement, defining the SQL request buffer and initializing the SPAREA, CSAINDX defines the attach, execution, and detach routines for the transfer in 5000-TRANSFER-PROCESS:

```
*****************************************************************
 * CONTROL THE PROCESS OF ATTACH, EXEC, DETATCH FOR TRANSFER.
 * 1) 1ST ATTACH TO DIRECTConnect CHECKS IF IT'S ALIVE AND WELL.
 * 2) 2ND ATTACH TO SQL SERVER - RUNS S.P. TO DELETE INDEXES.
 * 3) 3RD ATTACH TO DIRECTConnect RUNS TRANSFER FROM RSP TO SYBASE SQL SERVER.
 * 4) 4TH ATTACH TO SQL SERVER - RUNS S.P. TO RE-CREATE INDEXES.
 *****************************************************************
5000-TRANSFER-PROCESS.
    PERFORM 5100-WRITE-RUN-COUNT       THRU 5100-EXIT.
    PERFORM 5600-ATTACH-TO-DIRECTConnect THRU 5600-EXIT.
     PERFORM 5800-CALL-DETACH           THRU 5800-EXIT.
    PERFORM 5700-ATTACH-TO-SYBASE BASE SQL SERVER THRU 5700-EXIT.
     PERFORM 5200-LOAD-DEL-INDEX-STMT   THRU 5200-EXIT.
     PERFORM 5500-CALL-REQEXEC          THRU 5500-EXIT.
     PERFORM 5800-CALL-DETACH         THRU 5800-EXIT.
    PERFORM 5600-ATTACH-TO-DIRECTConnect THRU 5600-EXIT.
     PERFORM 5300-LOAD-TRANSFER-STMT  THRU 5300-EXIT.
     PERFORM 5500-CALL-REQEXEC        THRU 5500-EXIT.
     PERFORM 5800-CALL-DETACH         THRU 5800-EXIT.
    PERFORM 5700-ATTACH-TO-SYBASE BASE SQL SERVER THRU 5700-EXIT.
     PERFORM 5400-LOAD-CRE-INDEX-STMT THRU 5400-EXIT.
     PERFORM 5500-CALL-REQEXEC        THRU 5500-EXIT.
     PERFORM 5800-CALL-DETACH         THRU 5800-EXIT.
5000-EXIT.
     EXIT.
```

You can see that CSAINDX loads the TRANSFER request into the buffer and executes it.

```
                    5300-LOAD-STMT
*****************************************************************
 * LOAD THE TRANSFER STATEMENT INTO THE SQL BUFFER FOR THE DIRECTConnect
 *****************************************************************
5300-LOAD-TRANSFER-STMT.
    MOVE TRANSFER-STMT                          TO SQL-REQUEST.
     MOVE LENGTH OF TRANSFER-STMT                TO SQL-LENGTH.
     PERFORM 5500-CALL-REQEXEC                       THRU 5500-EXIT.
5300-EXIT.
```

```
EXIT.
```

Specifying error handling

CSAINDX handles status and messages in the same way that CLIENTC2 does. See “Understanding how CSAs handle errors” on page 24 for details. You should always use the CSA RESCHECK and GETMSG commands to determine if there are SQL request errors and to retrieve and log those that do occur.

APPENDIX D

CSARESCK Sample CSA

This appendix discusses the following topics:

- Downloading data: about the CSARESCK sample code
- CSARESCK sample code

Downloading data: about the CSARESCK sample code

The CSARESCK sample CSA downloads data to Adaptive Server Enterprise through a series of insert statements. It illustrates the error checking you need to include in a CSA when downloading data from a source on the mainframe directly to Adaptive Server Enterprise. CSARESCK shows you how to use the CSA RESCHECK and GETMSG commands to ensure that you download data correctly.

The Adaptive Server Enterprise table is specified with a unique index. Three of the statements fail with warnings because they attempt to place a value in a field that is already indexed with that value.

The duplicate insert statements are highlighted in the code.

It is important for you to understand that this method for determining which SQL statement succeeds and which fails works *only* if the error caused by the batched statements is a *warning level error*. In the case of warning messages, Adaptive Server Enterprise returns a success or warning message for each of the batched commands.

Note If any one of your batched commands causes a *serious error*, Adaptive Server Enterprise issues a ROLLBACK transaction, rolls back all the batched commands, and issues only one error message. This makes it impossible to determine the batched command that failed.

CSARESCK illustrates:

- Specifying an attachment definition name

- Using an SPAREA definition
- Setting up a request buffer and executing multiple insert statements
- Using the RESCHECK and GETMSG commands to ensure the SQL requests are successful in handling errors

CSARESCK sample code

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  CSARESCK.
*****************************************************************
 * SAMPLE CSA PROGRAM TO ILLUSTRATE ESTABLISHING A
 * ConnectION TO SQL SERVER, EXECUTING A SQL REQUEST THAT HAS
 * SEVERAL INSERT STATEMENTS - ONE OF WHICH IS BAD - AND THEN
 * SCROLLING THROUGH THE GETMSG AND RESCHECK FEATURES TO DETER-
 * MINE WHICH SQL COMMAND(S) RECEIVED AN ERROR.
 *
 * NOTE: THIS APPROACH FOR BATCHING SQL COMMANDS ONLY WORKS IN
 * SITUATIONS WHERE SQL SERVER DOES NOT RECEIVE AN ERROR SEVERE
 * ENOUGH TO CAUSE A ROLLBACK TRANSACTION (IN WHICH CASE THE CSA
 * WILL ONLY RECEIVE ONE ERROR MESSAGE FOR THE ENTIRE BATCH OF
 * COMMANDS).  IF A ROLLBACK TRANSACTION DOES NOT OCCUR, THEN
 * SQL SERVER WILL RETURN A SUCCESS/FAILURE MESSAGE FOR EACH OF
 * THE BATCHED SQL COMMANDS.
 *
 * TRANSID IN PCT: CRES         PROGRAM NAME IN PPT: CSARESCK
 *****************************************************************
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
*****************************************************************
 * POINTERS.
 *****************************************************************
01 WS-POINTERS.
     05 SPAREA-PTR                       POINTER.
     05 SQLDA-PTR                        POINTER.
     05 SQL-REQ-PTR                      POINTER.
*****************************************************************
 * COUNTERS AND VARIOUS INTEGERS.
 *****************************************************************
01 WS-VARIABLES.
     05 CICSRC                           PIC S9(8) COMP VALUE +0.
```

```
     05 RESCHECK-CNT                        PIC 9(3) VALUE ZEROES.
     05 SPSTATUS-CNT                        PIC 9(3) VALUE ZEROES.
*****************************************************************
 * ATTACHMENT DEFINITION NAME.
 *****************************************************************
O1 ATTACH-NAME                            PIC X(08) VALUE 'MDIAWONG'.
*****************************************************************
 * SWITCH FOR RESCHECK READS -
 *   THE IDEA IS TO KEEP CALLING RESCHECK UNTIL YOU'VE RECEIVED
 *   SPACES IN SPSTATUS TWICE IN A ROW - THEN YOU'RE DONE.
 *****************************************************************
01 WS-SWITCHES.
     03 WS-RESCHECK-DONE-SW                 PIC X VALUE 'N'.
         88 RESCHECK-DONE                   VALUE 'Y'.
         88 RESCHECK-NOT-DONE               VALUE 'N'.
         88 LAST-SPSTATUS-SPACES            VALUE ' '.
*****************************************************************
 * SQL STATEMENT TO EXECUTE.
 *****************************************************************
01 MULTI-INSERT-STMT.
     03 INSERT-1.
         05 FILLER                      PIC X(30) VALUE
         'INSERT INTO TESTABLE  VALUES ('.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(03) VALUE '001'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(02) VALUE ', '.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(16) VALUE
         'RECORD NUMBER 01'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(02) VALUE ', '.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(10) VALUE
         'FIRST RECD'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(04) VALUE ')   '.
         05 FILLER                      PIC X(1975) VALUE SPACES.
    03 INSERT-2.
         05 FILLER                      PIC X(30) VALUE
         'INSERT INTO TESTABLE  VALUES ('.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(03) VALUE '002'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(02) VALUE ', '.
         05 FILLER                      PIC X(01) VALUE QUOTE.
```

```
     05 FILLER                    PIC X(16) VALUE
     'RECORD NUMBER 02'.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(02) VALUE ', '.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(10) VALUE
     'SECOND REC'.
     05 FILLE                     PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(04) VALUE ') ; '.
03 INSERT-3-DUP.
     05 FILLER                    PIC X(30) VALUE
     'INSERT INTO TESTABLE  VALUES ('.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(03) VALUE '002'.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(02) VALUE ', '.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(16) VALUE
     'RECORD NUMBER 03'.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(02) VALUE ', '.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(10) VALUE
     'THIRD RECD'.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(04) VALUE ') ; '.
03 INSERT-4.
     05 FILLER                    PIC X(30) VALUE
     'INSERT INTO TESTABLE  VALUES ('.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(03) VALUE '004'.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(02) VALUE ', '.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(16) VALUE
     'RECORD NUMBER 04'.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(02) VALUE ', '.
     05 FILLER                    PIC X(01) VALUE QUOTE.
     05 FILLER                    PIC X(10) VALUE
     'FOURTH REC'.
     '05 FILLER                    PIC X(01) VALUE QUOTE.
     '05 FILLER                    PIC X(04) VALUE ') ; '.
03 INSERT-5-DUP.
     05 FILLER                    PIC X(30) VALUE
     'INSERT INTO TESTABLE  VALUES ('.
```

```
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(03) VALUE '004'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(02) VALUE ', '.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(16) VALUE
      'RECORD NUMBER 05'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(02) VALUE ', '.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(10) VALUE
      'FIFTH RECD'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(04) VALUE ') ; '.
03  INSERT-6.
      05 FILLER                    PIC X(30) VALUE
      'INSERT INTO TESTABLE  VALUES ('.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(03) VALUE '006'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(02) VALUE ', '.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(16) VALUE
      'RECORD NUMBER 06'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(02) VALUE ', '.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(10) VALUE
      'SIXTH RECD'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(04) VALUE ') ; '.
03 INSERT-7-DUP.
      05 FILLER                    PIC X(30) VALUE
      'INSERT INTO TESTABLE  VALUES ('.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(03) VALUE '006'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(02) VALUE ', '.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(16) VALUE
      'RECORD NUMBER 07'.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(02) VALUE ', '.
      05 FILLER                    PIC X(01) VALUE QUOTE.
      05 FILLER                    PIC X(10) VALUE
      '7TH   RECD'.
```

```
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(04) VALUE ') ; '.
    03  INSERT-8.
         05 FILLER                      PIC X(30) VALUE
         'INSERT INTO TESTABLE  VALUES ('.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(03) VALUE '008'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(02) VALUE ', '.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(16) VALUE
         'RECORD NUMBER 08'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(02) VALUE ', '.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(10) VALUE
         'LAST RECD'.
         05 FILLER                      PIC X(01) VALUE QUOTE.
         05 FILLER                      PIC X(04) VALUE ') ; '.
        ****************************************************************
 * FLAGS.
 ****************************************************************
01 WS-INIT-SPAREA-SW                  PIC X(01) VALUE '0'.
     88 INIT-SPAREA-OK                 VALUE '1'.
     88 INIT-SPAREA-BAD                VALUE '0'.
01 WS-ATTACH-SW                       PIC X(01) VALUE '0'.
     88 ATTACH-OK                      VALUE '1'.
     88 ATTACH-BAD                     VALUE '0'.
****************************************************************
 * ERROR MESSAGES.
 ****************************************************************
01 WS-STUB-ERROR-MSG.
     03 FILLER                         PIC X(09)
         VALUE 'CALL TO: '
     03 WS-STUB-NAME                   PIC X(08) VALUE SPACES.
     03 FILLER                         PIC X(18)
          VALUE ' - RECEIVED SPRC: '
     03 WS-STUB-SPRC                   PIC X(03) VALUE '000'.
     03 FILLER                         PIC X(03) VALUE ' - '.
     03 WS-STUB-SPMSG                  PIC X(100) VALUE SPACES.
01 WS-RESCHECK-LAST-MSG.
    03 FILLER                         PIC X(30)
         VALUE '- FINAL RESCHECK READ COUNT : '.
     03 WS-RESCHECK-COUNT              PIC 9(03) VALUE ZEROES.
01  WS-RESCHECK-MSG.
     03 FILLER                         PIC X(33)
```

```
         VALUE 'RESCHECK NON-BLANK STATUS - REC: '.
     03 WS-RESCHECK-NUMBER            PIC 9(03) VALUE ZEROES.
     03 FILLER                        PIC X(16)
         VALUE ' - SPSTATUS IS: '.
     03 WS-SPSTATUS-OUT               PIC X(02) VALUE SPACES.
     03 FILLER                        PIC X(11)
         VALUE ' - SPCODE: '.
     03 WS-SPCODE-OUT                 PIC X(03) VALUE SPACES.
     03 FILLER                        PIC X(09)
         VALUE ' - SPRC: '.
     03 WS-SPRC-OUT                   PIC X(03) VALUE SPACE.
     03 FILLER                        PIC X(10)
         VALUE ' - SPIND: '.
     03 WS-SPIND-OUT                  PIC X(01) VALUE SPACE.
LINKAGE SECTION.
01 STORED-PROC-AREA.
     COPY SPAREAC.
*****************************************************************
 * SQL REQUEST BUFFER THAT WILL BE PASSED TO THE REMOTE SERVER VIA
 * REQEXEC CALL.  IT CONSISTS OF A HALFWORD LENGTH FIELD, AND THE
 * ACTUAL REQUEST STATEMENT.
 *****************************************************************
 01 SQL-BUFFER.
     03 SQL-LENGTH                    PIC S9(4) COMP.
     03 SQL-REQUEST                   PIC X(3000).
*===============================================================*
 PROCEDURE DIVISION.
 *===============================================================*
0000-MAIN-PROCESSING.
    PERFORM 1000-SPAREA-INIT.
    IF INIT-SPAREA-OK
         PERFORM 2000-ATTACH-TO-SERVER.
         PERFORM 5000-EXEC-SQL-REQUEST
         PERFORM 6000-RESCHECK-REARCH
             UNTIL RESCHECK-DONE
         PERFORM 7000-CLOSE-DETACH
         PERFORM 9000-FINAL-COUNT
     END-IF.
0000-GET-OUT-NOW.
    EXEC CICS
         RETURN
     END-EXEC.
0000-EXIT.
     EXIT.
*****************************************************************
 * GET AN SPAREA, AND CALL CSA TO INITIALIZE IT.
```

```
 ****************************************************************
1000-SPAREA-INIT.
    EXEC CICS
         DELETEQ TS QUEUE('CSEXQUE2') RESP(CICSRC)
     END-EXEC.
    EXEC CICS
         GETMAIN SET(SPAREA-PTR)
         LENGTH(LENGTH OF SPAREA)
         NOSUSPEND
         RESP(CICSRC)
     END-EXEC.
    IF CICSRC = DFHRESP(NORMAL)
         SET ADDRESS OF STORED-PROC-AREA TO SPAREA-PTR
         MOVE 'Y' TO SPTRCOPT
         CALL 'CSSETUP' USING SPAREA
         IF SPRC = '000'
             SET INIT-SPAREA-OK       TO TRUE
         ELSE
             MOVE 'CSSETUP'           TO WS-STUB-NAME
             PERFORM 6900-STUB-CALL-ERROR THRU 6900-EXIT
             GO TO 0000-GET-OUT-NOW
         END-IF
     END-IF.
1000-EXIT.
         EXIT.
****************************************************************
 * CALL CSA TO ATTACH TO REMOTE SERVER.
 ****************************************************************
2000-ATTACH-TO-SERVER.
    MOVE ATTACH-NAME                TO SPATTACH.
    CALL 'ATTACH' USING SPAREA.
    IF SPRC = '000'
         SET ATTACH-OK               TO TRUE
     ELSE
         MOVE 'ATTACH'               TO WS-STUB-NAME
         PERFORM 6900-STUB-CALL-ERROR THRU 6900-EXIT
         GO TO 0000-GET-OUT-NOW
     END-IF.
2000-EXIT.
         EXIT.
****************************************************************
 * EXECUTE THE SQL REQUEST AGAINST THE REMOTE SERVER.
 ****************************************************************
5000-EXEC-SQL-REQUEST.
    EXEC CICS
         GETMAIN SET(SQL-REQ-PTR)
```

```
         LENGTH(LENGTH OF SQL-BUFFER)
         NOSUSPEND RESP(CICSRC)
     END-EXEC.
    SET ADDRESS OF SQL-BUFFER        TO SQL-REQ-PTR.
     SET SPSQL TO SQL-REQ-PTR.
    IF CICSRC = DFHRESP(NORMAL)
         MOVE MULTI-INSERT-STMT       TO SQL-REQUEST
         MOVE LENGTH OF MULTI-INSERT-STMT TO SQL-LENGTH
         CALL 'REQEXEC' USING SPAREA
     END-IF.
    5000-EXIT.
         EXIT.
*****************************************************************
 * CHECK RESCHECK TO LOCATE SQL STATEMENT IN ERROR.
 *****************************************************************
6000-RESCHECK-REARCH.
    ADD 1                            TO RESCHECK-CNT.
    IF SPSTATUS NOT = '  '
         MOVE 'N'                     TO WS-RESCHECK-DONE-SW
         ADD 1                        TO SPSTATUS-CNT
         MOVE SPSTATUS-CNT            TO WS-RESCHECK-NUMBER
         MOVE SPSTATUS                TO WS-SPSTATUS-OUT
         MOVE SPCODE                  TO WS-SPCODE-OUT
         MOVE SPRC                    TO WS-SPRC-OUT
         MOVE SPIND                   TO WS-SPIND-OUT
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE2')
                 FROM(WS-RESCHECK-MSG) NOSUSPEND
                 RESP(CICSRC)
         END-EXEC
         IF SPIND NOT = SPACES
             PERFORM 6500-GET-MESSAGES UNTIL SPIND NOT = 'M'
         END-IF
     ELSE
         IF LAST-SPSTATUS-SPACES
             MOVE 'Y'                 TO WS-RESCHECK-DONE-SW
         ELSE
             MOVE ' '                 TO WS-RESCHECK-DONE-SW
         END-IF
     END-IF.


    IF NOT RESCHECK-DONE
         CALL 'RESCHECK' USING SPAREA.
```

```
6000-EXIT.
     EXIT.
*****************************************************************
* RETRIEVE ANY OUTSTANDING MESSAGES FOR A REQUEST.
*****************************************************************
6500-GET-MESSAGES.
    CALL 'GETMSG' USING SPAREA.
     IF SPMSG NOT = SPACES
         EXEC CICS
             WRITEQ TS QUEUE('CSEXQUE2')
                 FROM(SPMSG) NOSUSPEND
             RESP(CICSRC)
         END-EXEC
     END-IF.
6500-EXIT.
     EXIT.
*****************************************************************
 * FORMAT AND WRITE STUB-CALL ERROR INFO TO TS QUEUE.
 *****************************************************************
6900-STUB-CALL-ERROR.
    MOVE SPRC                        TO WS-STUB-SPRC.
     MOVE SPMSG                       TO WS-STUB-SPMSG.
    EXEC CICS
         WRITEQ TS QUEUE('CSEXQUE2')
             FROM(WS-STUB-ERROR-MSG) NOSUSPEND
         RESP(CICSRC)
     END-EXEC.
6900-EXIT.
     EXIT.
*****************************************************************
 * FORMAT AND WRITE STUB-CALL ERROR INFO TO TS QUEUE.
 *****************************************************************
7000-CLOSE-DETACH.
        CALL 'DETACH'     USING SPAREA.
    IF SPRC NOT = '000'
         MOVE 'DETACH'                TO WS-STUB-NAME
         PERFORM 6900-STUB-CALL-ERROR THRU 6900-EXIT
         GO TO 0000-GET-OUT-NOW
     END-IF.
7000-EXIT.
     EXIT.
*****************************************************************
 * SEND THE FINAL RESCHECK READ NUMBER TO TEMP STORAGE QUE
 *****************************************************************
9000-FINAL-COUNT.
    MOVE RESCHECK-CNT                TO WS-RESCHECK-COUNT.
```

```
    EXEC CICS
         WRITEQ TS QUEUE('CSEXQUE2')
             FROM(WS-RESCHECK-LAST-MSG) NOSUSPEND
             RESP(CICSRC)
     END-EXEC.
9000-EXIT.
     EXIT.
*================================================================*
 *    END OF PROGRAM.
 *================================================================*
```

APPENDIX E

The SPAREA

This appendix discusses the following topics:

- How CSAs use the SPAREA
- SPAREA field descriptions
- Copying SPAREA definitions to the CSA
- SPAREA definitions

How CSAs use the SPAREA

A CSA exchanges information with Open ClientConnect by retrieving from and placing values to fields in the Stored Procedure Communication Area (SPAREA).

The CSA establishes the SPAREA for Open ClientConnect with the CSSETUP command. When the CSA issues another command (for example, OPENPIPE), this area communicates the command parameters to Open ClientConnect. Open ClientConnect uses the SPAREA to communicate the results of the command back to the CSA. For information on CSA commands, see Appendix A, “CSA Commands.”

Note An SPAREA definition must be included in every CSA. Copy library members that describe the SPAREA for each supported language are included on the Open ClientConnect API and are also described at the end of this chapter.

SPAREA field descriptions

The CSA and Open ClientConnect use the SPAREA by putting values into and retrieving values from the SPAREA fields to communicate. The word *Reserved* in the descriptions indicates that the CSA cannot write to the field.

SPHEADER SPHEADER contains the character string *SPAREA*. The character string serves as an eye catcher for locating the SPAREA in a dump. *Reserved.*

SPRESRVD SPRESRVD contains values used by Open ClientConnect to process commands. *Reserved*.

SPSTATUS Open ClientConnect uses SPSTATUS to indicate the success or failure of processing. This value refers to processing on the remote database. Valid values are:

- 'OK' indicates success.
- 'E' indicates an error.
- 'W' indicates a warning.
- 'R' indicates results.

SPCODE Only RSPs use SPCODE. *Reserved.*

SPFORMAT The CSA uses SPFORMAT to specify the data format when opening a data pipe. The only valid value for CSAs is DB2.

SPMODE The CSA uses SPMODE to specify the mode of the data pipe. The only valid value for CSAs is INPUT.

SPRC Open ClientConnect uses SPRC to indicate the success or failure of a CSA command. Valid return codes are:

- '000' indicates successful completion.
- *'xxx'* indicates an Open ClientConnect error number.
- 'EOF' indicates an End of File on input data.

SPSQLDA Open ClientConnect uses SPSQLDA to specify the address of a SQLDA that describes the data records. For details, see *IBM SQL Reference for DB2*.

SPROWS SPROWS contains the number of rows returned for CSA requests that return data.

SPPREFIX Open ClientConnect uses SPPREFIX to provide messages with an appropriate message prefix. *Reserved*.

SPMSG	Open ClientConnect uses SPMSG to place message text to be retrieved by the CSA with a GETMSG command.
SPFILL2	SPFILL2 is used for slack bytes in the SPAREA. *Reserved.*
SPSQL	The CSA uses SPSQL to specify a pointer to the SQL request buffer containing the SQL statements it is sending to Open ClientConnect.
SPSERVER	The CSA uses SPSERVER to provide the name of the LAN-based server on the remote platform to which the CSA connects. The CSA must provide the SPSERVER or the SPATTACH before issuing the ATTACH command.
SPATTACH	The CSA uses SPATTACH to provide the name of the Sybase attachment definition to connect to the remote platform for backward compatibility. The CSA must provide SPATTACH or SPSERVER before issuing the ATTACH command. If the attachment definition is specified but not found in the *PCSQLSP* file, the attachment name is used as the server name.
SPUSERID	The CSA uses SPUSERID to specify a user ID required by the remote database (for example, Adaptive Server Enterprise, DB2, or DBM). For backward compatibility, if the CSA does not specify SPUSERID and a Sybase attachment name is used and found in the *PCSQLSP* file, the CSA uses the user ID in the attachment definition is used.
SPPWD	The CSA uses SPPWD to specify a password required by the remote database. For backward compatibility, if the CSA does not specify SPPWD and a Sybase attachment name is used and found in the *PCSQLSP* file, the CSA uses the password in the attachment definition.
SPCMPOPT	SPCMPOPT is no longer used.
SPIND	Open ClientConnect uses SPIND to inform a CSA that an additional message is available. Valid values are: • M indicates that messages are available. • [Blank] indicates that messages are not available.
SPDATE	Open ClientConnect uses SPDATE to inform a CSA of the date that a request ran. This field contains a value in the form *YY/MM/DD*.
SPTIME	Open ClientConnect uses SPTIME to inform a CSA of the time that a request ran. This field contains a value in the form *HH:MM:SS*.
SPCONFIG	SPCONFIG is no longer used.

SPEND The assembler programming language uses SPEND as a label indicating the end of SPAREA. COBOL II, C, and PL/1 SPAREA descriptions do not include SPEND.

SPTRCOPT SPTRCOPT controls the trace option. If this field contains 'Y' when an Open ClientConnect command is issued, trace records are written to the TSQ, CE*xxxxxx*, where *xxxxxx* is the first six characters of the user ID.

Copying SPAREA definitions to the CSA

Within your CSA, copy the SPAREA definition as shown in Table E-1. For an example of copying the SPAREA in the context of a CSA written in COBOL II, see the samples in Chapter 3, “Writing a CSA.”

Table E-1: SPAREA copy statements

Language	Copy syntax
Assembler	`COPY SPAREAA`
COBOL II	`COPY SPAREAC.`
PL/I	`EXEC SQL INCLUDE SPAREAP;`
C	`#include "SPAREAX.H"`

When you compile the CSA, the concatenation sequence for SYSLIB must include a DD statement for the Open ClientConnect sample program library. See Chapter 4, “Compiling and Testing a CSA” for details.

SPAREA definitions

SPAREA definitions in Assembler, COBOL II, PL/I, and C are distributed with Open ServerConnect and are reproduced in this appendix.

The following MVS SPAREA definitions are in the SYBASE.0CSA310B.CICS.LOADLIB, and their definitions are reproduced on the following pages.

Note Several fields in the SPAREA definitions are used only for **RSPs**. Those fields are described in the Mainframe Connect Server Option *Programmer's Reference for Remote Stored Procedures*.

SPAREAA assembler definition

```
*---------------------------------------*
 *  STORED PROCEDURE COMMUNICATION AREA  *
 *---------------------------------------*
03  SPAREA.
 05  SPHEADER          PIC X(8).
     05  SPRESRVD          PIC X(33).
     05  SPTRCOPT          PIC X.
     05  SPSTATUS          PIC X(2).
     05  SPCODE            PIC X(8).
     05  SPFORMAT          PIC X(3).
     05  SPMODE            PIC X(6).
     05  SPRC              PIC X(3).
     05  SPFROM            USAGE IS POINTER.
     05  SPINTO            REDEFINES SPFROM USAGE IS POINTER.
     05  SPSQLDA           REDEFINES SPINTO USAGE IS POINTER.
     05  SPVARTXT          USAGE IS POINTER.
     05  SPVARTAB          USAGE IS POINTER.
     05  SPROWS            PIC S9(8) COMP.
     05  SPMAXLEN          PIC S9(4) COMP.
     05  SPRECLEN          REDEFINES SPMAXLEN PIC S9(4) COMP.
     05  SPVARLEN          PIC S9(4) COMP.
     05  SPPREFIX          PIC X.
     05  SPMSG             PIC X(100).
     05  FILLER            PIC X(3).
     05  SPSQL             USAGE IS POINTER.
     05  SPATTACH          PIC X(8).
     05  SPUSERID          PIC X(8).
     05  SPPWD             PIC X(8).
     05  SPCMPOPT          PIC X(1).
     05  SPIND             PIC X(1).
     05  SPDATE            PIC X(8).
     05  SPTIME            PIC X(8).
     05  SPCONFIG          PIC X(4).
```

```
     05  SPSERVER          PIC X(30).
     05  FILLER            PIC X(32).
```

SPAREAC COBOL II definition

```
*---------------------------------------*
 *  STORED PROCEDURE COMMUNICATION AREA  *
 *---------------------------------------*
SPAREA   DSECT
SPHEADER DS    CL8                     EYE CATCHER
 SPRESRVD DS    CL33                    SERVER INFORMATION
 SPTRCOPT DS    CL1                     TRACE OPTION
 SPSTATUS DS    CL2                     STATUS INDICATOR
 SPCODE   DS    CL8                     ERROR CODE
 SPFORMAT DS    CL3                     PIPE FORMAT
 SPMODE   DS    CL6                     PIPE MODE
 SPRC     DS    CL3                     RETURN CODE
 SPFROM   DS    0F                      FROM ADDRESS
 SPINTO   DS    0F                      INTO ADDRESS
 SPSQLDA  DS    F                       SQLDA ADDRESS
 SPVARTXT DS    F                       VARIABLE TEXT
 SPVARTAB DS    F                       VARIABLE TABLE
 SPROWS   DS    F                       ROWS AFFECTED
 SPMAXLEN DS    0H                      MAXIMUM LENGTH OF STD RECORD
 SPRECLEN DS    H                       RECORD LENGTH
 SPVARLEN DS    H                       VARIABLE TEXT LENGTH
 SPPREFIX DS    CL1                     MESSAGE FILE PREFIX
 SPMSG    DS    CL100                   MESSAGE AREA
 SPFILL2  DS    CL3                     NOT USED
 SPSQL    DS    F                       SQL BUFFER ADDRESS
 SPATTACH DS    CL8                    SERVER NAME, ID and PSWD
 SPUSERID DS    CL8                     USERID
 SPPWD    DS    CL8                     PASSWORD
 SPCMPOPT DS    CL1                     COMPRESSION OPTION
 SPIND    DS    CL1                     MESSAGE INDICATOR
 SPDATE   DS    CL8                     DATE
 SPTIME   DS    CL8                     TIME
 SPCONFIG DS    CL4                     CONFIGURATION ID
 SPSERVER DS    CL30  SERVER NAME (SNA = SYMS, TCP/IP = SygwHosts
          DS    CL32                    FILLER
SPEND    EQU   *
```

SPAREAP PL/1 definition

```
/*************************************/
 /* STORED PROCEDURE COMMUNICATION AREA  */
 /*************************************/
DCL 1 COMMPTR                POINTER;
 DCL 1 SPAREA BASED(COMMPTR),
       3  SPHEADER            CHAR(8),
       3  SPRESRVD            CHAR(33),
       3  SPTRCOPT            CHAR(1),
       3  SPSTATUS            CHAR(2),
       3  SPCODE              CHAR(8),
       3  SPFORMAT            CHAR(3),
       3  SPMODE              CHAR(6),
       3  SPRC                CHAR(3),
       3  SPFROM              POINTER ALIGNED,
       3  SPVARTXT            POINTER,
       3  SPVARTAB            POINTER,
       3  SPROWS              FIXED BIN(31) ALIGNED,
       3  SPMAXLEN            FIXED BIN(15) ALIGNED,
       3  SPVARLEN            FIXED BIN(15) ALIGNED,
       3  SPPREFIX            CHAR,
       3  SPMSG               CHAR(100),
       3  SPFILL2             CHAR(3),
       3  SPSQL               POINTER ALIGNED,
       3  SPATTACH            CHAR(8),
       3  SPUSERID            CHAR(8),
       3  SPPWD               CHAR(8),
       3  SPCMPOPT            CHAR(1),
       3  SPIND               CHAR(1),
       3  SPDATE              CHAR(8),
       3  SPTIME              CHAR(8);
       3  SPCONFIG            CHAR(4);
       3  SPSERVER            CHAR(30);
       3  SPFILL3             CHAR(32);
 DCL SPINTOPOINTER BASED(AD_SPFROM);
 DCL SPSQLDAPOINTER BASED(AD_SPFROM);
 DCL SPRECLEN POINTER BASED(AD_SPMAXLEN);
 DCL SPSQLPOINTER BASED(AD_SPSQL);
 DCL (AD_SPFROM, AD_SPMAXLEN, AD_SPSQL) POINTER;
 AD_SPFROM=ADDR(SPFROM);
 AD_SPMAXLEN=ADDR(SPMAXLEN);
 AD_SPSQL=ADDR(SPSQL);
```

SPAREAX C definition

```
#ifndef SP_DEFS
 #define SP_DEFS
 /*
    Various declarations and definitions for Stored Procedures for C.
    Should be usable with the SAS/C compiler, and with slight
    modification, the IBM C/370 compiler.  Uses the SAS/C digraphs for
    square brackets - "(]" for the left square bracket, and "])" for the
    right square bracket.
   SAS/C and C/370 are trademarks of the SAS Institute, Inc. and IBM
    Corporation respectively.
*/
 #include "sqlda.h"
/*
    Keyword variable table declaration.
 */
struct VARTAB {
   unsigned long varTabL;    /* Number of entries in table (<<= 50)   */
   struct VARENT {
     char *varName;          /* Variable name                         */
     char *varValue;         /* Variable value                        */
     short varNameL;         /* Variable name length                  */
     short varValL;          /* Variable value length                 */
   } varent(]50]);
 };
/*
    Stored Procedure Communication Area declaration.
 */
 struct SPAREA {
   char spheader(]8]);       /* DS    CL8   Eye catcher               */
   char spresrvd(]33]);      /* DS    CL33  Server information        */
   char sptrcopt;            /* DS    CL1   Trace option              */
   char spstatus(]2]);       /* DS    CL2   Status indicator          */
   char spcode(]8]);         /* DS    CL8   Error code                */
   char spformat(]3]);       /* DS    CL3   Pipe format               */
   char spmode(]6]);         /* DS    CL6   Pipe mode                 */
   char sprc(]3]);           /* DS    CL3   Return code               */
   union {
     char *spfrom;           /* DS    0A    From address              */
     char *spinto;           /* DS    0A    Into address              */
     struct SQLDA *spsqlda;  /* DS    A     SQLDA address             */
   };
   char *spvartxt;           /* DS    A     Variable text             */
   struct VARTAB *spvartab;  /* DS    A     Variable table            */
   int sprows;               /* DS    F     Rows affected             */
   union {
```

```
    short spmaxlen;          /* DS    0H    Max length of STD rec    */
 short spreclen;    /* DS    H     Record length            */
 };
 short spvarlen;           /* DS    H     Variable text length     */
 char spprefix;            /* DS    CL1   Message file prefix      */
 char spmsg(]100]);        /* DS    CL100 Message area             */
 char _f0(]3]);            /* Padding for alignment                */
 struct SQLBUF *spsql;     /* DS    A     SQL buffer address       */
 char spattach(]8]);       /* DS    CL8   Attachment name          */
 char spuserid(]8]);       /* DS    CL8   Userid                   */
 char sppwd(]8]);          /* DS    CL8   Password                 */
 char spcmpopt;            /* DS    CL1   Compression option       */
 char spind;               /* DS    CL1   Message indicator        */
 char spdate(]8]);         /* DS    CL8   Request execution date   */
 char sptime(]8]);         /* DS    CL8   Request execution time   */
 char spconfig(]4]); /* DS  CL4 Configuration name       */
 char spserver(]30]);      /*DS     CL30  Server name    */
 char _f1(]32]);           /* Padding to end of record             */
 };
/*
    Stored procedure function declarations.
 */
 void attach(struct SPAREA *);         /* Attach to remote server     */
 void clospipe(struct SPAREA *);       /* Close input/output pipe     */
 void commit(struct SPAREA *);         /* Issue SYNCPOINT w/COMMIT    */
 void cssetup(struct SPAREA *);        /* Initialize SPAREA           */
 void detach(struct SPAREA *);         /* Detach from remote server   */
 void getmsg(struct SPAREA *);         /* Get a message               */
 void getpipe(struct SPAREA *);        /* Get row from input pipe     */
 void getpipe(struct SPAREA *);        /* Put row to output pipe      */
 void message(struct SPAREA *);        /* Issue message               */
 void openpipe(struct SPAREA *);       /* Open input/output pipe      */
 void reqexec(struct SPAREA *);        /* Execute SQL request         */
 void rescheck(struct SPAREA *);       /* Check for results           */
 void rollback(struct SPAREA *);       /* Issue SYNCPOINT w/ROLLBACK  */
 void status(struct SPAREA *);         /* Issue status                */
#endif
```

APPENDIX F

The SQLDA

This appendix discusses the following topics:

- How CSAs use the SQLDA
- SQLDA variables and fields
- SQLDA Datatypes
- Sample COBOL-language SQLDA description
- Sample C-language SQLDA description

How CSAs use the SQLDA

The SQLDA is a standard data structure that defines a multi-column file. All files are exchanged between the CSA and Open ClientConnect using the SQLDA.

This appendix provides sample COBOL-language and C-language SQLDA declarations for DB2 datatypes. See the *IBM DB2 SQL Reference* for information on the SQLDA.

SQLDA variables and fields

A SQLDA consists of four variables (*SQLDAID*, *SQLDABC*, *SQLN*, and *SQLD*), followed by an arbitrary number of SQLVARs. A SQLVAR is a structure containing five fields.

The following table describes the SQLDA variables.

Table F-1: SQLDA variables

This SQLDA variable:	Performs this function:
SQLDAID	Contains an eye catcher of "SQLDA" for use in storage dumps
SQLDABC	Contains the length of the SQLDA, equal to SQLN*44+16
SQLN	Contains the total number of occurrences of SQLVAR
SQLD	Indicates the number of columns described by occurrences of SQLVAR

Each occurrence of SQLVAR describes one column of the result row the CSA is sending to the client application. The following table describes the five fields that each occurrence of SQLVAR contains.

Table F-2: SQLDA fields

This SQLDA field:	Performs this function:
SQLTYPE	Contains a 3-digit value that represents the datatype of the column and whether or not it allows null values. Table F-3 on page 100 contains the valid data type values.
SQLLEN	Contains the external length of a value from the column.
SQLDATA	Contains the address of the data being transmitted.
SQLIND	Contains the address of an indicator, which tells whether the column is nullable. Uses a value less than zero if null.
SQLNAME	Contains the name or label of the column, or a string of length zero if the name or label does not exist.
SQLNAMEL	Contains the length of the column.

Relating these standard SQLDA fields to the SQLDA template example from the LINKAGE SECTION of CLIENTC2 (see Appendix B, "CLIENTC2 Sample CSA"), you can see that one SQLVAR definition can be used six times for the six columns of data to be received from the SALES table:

```
01 SALES-SQLDA.
    03 SALES-SQLDAID     PIC X(08).
    03 SALES-SQLDABC     PIC S9(8) COMP.
    03 SALES-SQLN        PIC S9(4) COMP.
    03 SALES-SQLD        PIC S9(4) COMP.
    03 SALES-SQLVAR      OCCURS 6 TIMES.
       05 SALES-SQLTYPE  PIC S9(4) COMP.
       05 SALES-SQLLEN   PIC S9(4) COMP.
```

```
        05 SALES-SQLDATA  POINTER.
        05 SALES-SQLIND   POINTER.
        05 SALES-SQLNAME  PIC X(32).
```

The example assumes that all the expected data is the same datatype. If that were not true, you would need to define separate SQLVARs, with separate *SQLIND* fields for each of the datatypes in your SQLDA template.

In each of the six SQLVARs, a *SALES-SQLDATA* field points to a result data field that the CSA expects to receive. In CLIENTC2, the result data fields are defined in the LINKAGE SECTION following the SQLDA template:

```
01 STORE-ID              PIC X(04).
01 ORDER-NUMBER.
     03 ORD-NUM-LENGTH   PIC S9(4) COMP.
     03 ORD-NUM.
        05 ORD-NUMCHAR   PIC X(01)
        OCCURS 20 TIMES DEPENDING ON ORD-NUM-LENGTH.
01 ORDER-DATE            PIC X(10).
01 QUANTITY              PIC S9(4) COMP.
01  PAY-TERMS.
    03 PAY-TERM-LEN      PIC S9(4) COMP.
    03 PAY-TERM.
        05 PAY-TERM-CHAR PIC X(01)
        OCCURS 12 TIMES DEPENDING ON PAY-TERM-LEN.
01  TITLE-ID-ENT.
     03 TITLE-ID-LEN     PIC S9(4) COMP.
     03 TITLE-ID.
        05 TITLE-ID-CHAR PIC X(01)
        OCCURS 6 TIMES DEPENDING ON TITLE-ID-LEN.
```

After the input pipe is open and Open ClientConnect passes the actual SQLDA, the pointers to the result data are related to the data field definitions:

```
SET ADDRESS OF SALES-SQLDA  TO SPSQLDA
SET ADDRESS OF STORE-ID     TO SALES-SQLDATA(1)
 SET ADDRESS OF ORDER-NUMBER TO SALES-SQLDATA(2)
 SET ADDRESS OF ORDER-DATE   TO SALES-SQLDATA(3)
 SET ADDRESS OF QUANTITY     TO SALES-SQLDATA(4)
 SET ADDRESS OF PAY-TERMS    TO SALES-SQLDATA(5)
 SET ADDRESS OF TITLE-ID-ENT TO SALES-SQLDATA(6)
```

CLIENTC2 illustrates the steps for reading a SQLDA definition:

1 Include a description of the SQLDA template.

  The SQLDA template and the result data definitions go in the LINKAGE SECTION so they can be accessed by programs outside the CSA, such as Open ClientConnect.

2 Set the address of the SQLDA template to the address of the SQLDA that Open ClientConnect sends using the SPSQLDA field. Open ClientConnect places that address in the SPSQLDA field of the SPAREA.

3 Relate the pointers to the result data field definitions.

SQLDA Datatypes

Table F-3 contains the SQLDA data types and their 3-digit values. Each data type has two available values to indicate whether an occurrence of the datatype allows nulls. Use the value that describes how nulls are handled.

Table F-3: SQLDA datatypes

Datatype	Value, no nulls	Value, nulls allowed
DATE	384	385
TIME	388	389
TIMESTAMP	392	393
CHAR VARIABLE LENG	448	449
CHAR FIXED LENGTH	452	453
CHAR LONG VARIABLE	456	457
FLOATING-POINT	480	481
DECIMAL	484	485
LARGE INTEGER	496	497
SMALL INTEGER	500	501

Sample COBOL-language SQLDA description

The following sample description of the SQLDA is for the COBOL II programming language. A complete description of each field and its purpose is in the *IBM DB2 SQL Reference*.

Note You can initializze SQLDABC (SQLDA Byte Count) with

```
MOVE LENGTH OF SQLDA TO SQLDABC
```

```
*****************************************************************
 * The following sample description of the SQLDA is for COBOL II.
 * A complete description of each field and its purpose is in
 * the "DB2 SQL Reference."  Note that SQLDABC (SQLDA
 * Byte Count) may be initialized with:
 *
 * MOVE LENGTH OF SQLDA TO SQLDABC.
 *****************************************************************
 01  SQLDA.
     03  SQLDAID                    PIC X(8).
     03  SQLDABC                    PIC S9(8) COMP.
     03  SQLN                       PIC S9(4) COMP.
     03  SQLD                       PIC S9(4) COMP.
     03  SQLVAR                     OCCURS 0 TO 300 TIMES
                                    DEPENDING ON SQLN.
         05  SQLTYPE               PIC S9(4) COMP.
         05  SQLLEN                PIC S9(4) COMP.
         05  SQLDATA               USAGE IS POINTER.
         05  SQLIND                USAGE IS POINTER.
         05  SQLNAME.
             07  SQLNAMELENGTH     PIC S9(4) COMP.
             07  SQLNAMEVALUE      PIC X(30).
```

Sample C-language SQLDA description

The following sample description of the SQLDA is for the C programming language. It provides a sample SQLDA definition and definition statements for all DB2 datatypes.

```
/*
   Sample SQLDA declaration and #defines for all DB2 datatypes.
 */
#ifndef SQLDA_DEF
 #define SQLDA_DEF
 struct SQLDA {
   unsigned char sqldaid[8];
   long sqldabc;
   short sqln;
   short sqld;
   struct sqlvar {
     short sqltype;
     union {
       short sqllen;
```

```
       struct {
         unsigned char precision;
         unsigned char scale;
       } SQLDECIMAL;
     } SQLLEN;
     unsigned char *sqldata;
     short *sqlind;
     struct sqlname {
       short length;
       unsigned char data [30];
     } sqlname;
 } sqlvar[0];
 };
#define DATE 384           /* SQLTYPE for DATE              */
 #define NDATE 385          /* SQLTYPE for DATE w/NULL       */
 #define TIME 388           /* SQLTYPE for TIME              */
 #define NTIME 389          /* SQLTYPE for TIME w/NULL       */
 #define TIMESTAMP 392      /* SQLTYPE for TIMESTAMP         */
 #define NTIMESTAMP 393     /* SQLTYPE for TIMESTAMP W/NULL  */
 #define VARCHAR 448        /* SQLTYPE for VARCHAR           */
 #define NVARCHAR 449       /* SQLTYPE for VARCHAR w/NULL    */
 #define CHAR 452           /* SQLTYPE for CVARCHAR          */
 #define NCHAR 453          /* SQLTYPE for VARCHAR w/NULL    */
 #define LONGVARCHAR 456    /* SQLTYPE for LONG VARCHAR      */
 #define NLONGVARCHAR 457   /* SQLTYPE for LVARCHAR w/ NULL  */
 #define FLOAT 480          /* SQLTYPE for FLOAT             */
 #define NFLOAT 481         /* SQLTYPE for FLOAT  w/ NULL    */
 #define DECIMAL 48         /* SQLTYPE for DECIMAL           */
 #define NDECIMAL 485       /* SQLTYPE for DECIMAL w/ NULLS  */
 #define INTEGER 496        /* SQLTYPE for INTEGER           */
 #define NINTEGER 497       /* SQLTYPE for INTEGER w/ NULL   */
 #define SMALLINT 500       /* SQLTYPE for SMALLINT          */
 #define NSMALLINT 501      /* SQLTYPE for SMALL w/ NULL     */
#endif
```

APPENDIX G

Related Products and Documentation by Component

This appendix includes the following topics:

- Related Sybase products
- Related IBM products
- Mainframe Connect documentation by component

Related Sybase products

Mainframe Connect components can be used with most Sybase products to connect mainframe applications with remote clients and servers. The products in Table G-1 are frequently used as clients and/or servers with the Mainframe Connect product set.

You need Sybase, IBM, and vendor communications documentation to configure, install, and manage Mainframe Connect products. You may need additional IBM and vendor manuals for your particular configuration.

Table G-1: Related Sybase products

Product	Description
Adaptive Server Enterprise	With Adaptive Server Enterprise acting as a server, clients using Open ClientConnect can send requests to a named Adaptive Server Enterprise through Mainframe Client Connect. Adaptive Server Enterprise, Sybase's RDBMS, can act as a client or server. Networked clients can call Adaptive Server Enterprise stored procedures that send requests to Open ServerConnect or MainframeConnect for DB2 UDB (or OmniSQL Access Module™ for™ DB2) on the mainframe. The DirectConnect administrator can configure the TRS to require all client requests to be routed through Adaptive Server Enterprise.

Product	Description
ASE/CIS (formerly OmniConnect™)	ASE/CIS translates SQL statements that DB2 cannot read or process into statements that DB2 can process. OmniSQL Access Module clients can route requests through OmniConnect for SQL translation. Sites that need to run the same applications without modifications against both Adaptive Server Enterprise and DB2 can use ASE/CIS as a server.
Open Client™	Open Client is a Sybase connectivity product that provides customer client applications, third-party products, and other Sybase products with the interfaces needed to communicate with Sybase servers and Open Server applications, including Open ServerConnect and OmniSQL Access Module™ for™ DB2 and MainframeConnect.
Open Server™	Open Server provides the tools and interfaces needed to create a custom server (called an Open Server application). Open ServerConnect uses Open Server architecture to allow the mainframe to act as a server. Open ClientConnect clients can send requests to Open Server applications on the local area network (LAN) through Mainframe Client Gateway, or directly through Intersystem Communication (ISC) if requests are being sent to Open ServerConnect in a separate region.
Replication Server®	Replication Server keeps copies of data up-to-date at multiple sites so that clients can access local data instead of remote, centralized databases.
Replication Agent®	Replication Agent is a functional component of the replication model that conveys updates made to primary data sources to a Replication Server.

Related IBM products

The Mainframe Access Products Set uses a number of IBM mainframe software products to allow mainframes to act as clients and servers. The most important of these products are listed in Table G-2.

Table G-2: Related IBM products

Product	Description
APPC/LU 6.2 (Advanced Program-to-Program Communication facility, which is a Logical Unit Type 6.2.)	IBM's program-to-program communication protocol. Open ServerConnect, Open ClientConnect, DB2 Access Module, and MainframeConnect for DB2 UDB can all use APPC to communicate with DirectConnect.
C/370	An API that is available in Open ClientConnect 3.2.

Product	Description
CICS (Customer Information Control System)	A transaction monitor on the mainframe. Open ServerConnect, Open ClientConnect, and MainframeConnect for DB2 UDB are all available for CICS.
DB2 (Database 2)	A relational database on the mainframe that uses SQL. Mainframe Access Products clients can send requests to DB2.
IMS TM (Information Management System Transaction Manager)	A transaction monitor and data communication system on the mainframe. Open ServerConnect, Open ClientConnect, and the OmniSQL Access Module™ for™DB2 are all available for IMS TM.
OS PL/I Version II	An application programming language. Open Client Client-Library and Open Server Gateway-Library are both available for PL/I.
TCP/IP (Transmission Control Protocol/Internet Protocol)	A communication protocol in general use. Open ServerConnect, Open ClientConnect, the OmniSQL Access Module™ for DB2, and MainframeConnect for DB2 UDB, can all use TCP/IP to communicate through DirectConnect.
VS COBOL II	An application programming language. Open Client Client-Library and Open Server Gateway-Library are both available for COBOL.
VTAM (Virtual Telecommunications Access Method)	A mainframe access method used with SNA. Mainframe-based Mainframe Access Products products run in a VTAM environment.

Mainframe Connect documentation by component

This section summarizes Mainframe Connect documentation by component.

Table G-3: Sybase documentation for Mainframe Connect products

If you have this product	Use these Sybase publications
Open ServerConnect	Mainframe Connect Server Option *Programmer's Reference* (a separate manual for each supported language)
	Mainframe Connect Server Option *Installation and Administration Guide* (for your mainframe system and network protocol)
	Mainframe Connect Server Option *Programmer's Reference for Remote Stored Procedures* (also used with MainframeConnect for DB2 UDB)
	Mainframe Connect Client Option and Server Option *Messages and Codes*
Open ClientConnect	Mainframe Connect Client Option *Programmer's Reference* (a separate manual for each supported language)
	Mainframe Connect Client Option *Installation and Administration Guide* (for your mainframe system and network protocol)
	Mainframe Connect Client Option *Programmer's Reference for Client Services Applications* (also used with MainframeConnect for DB2 UDB)
DirectConnect	Enterprise Connect Data Access and Mainframe Connect *Server Administration Guide* for DirectConnect
	Mainframe Connect DirectConnect for z/OS Option *Installation Guide*
	Mainframe Connect DirectConnect for z/OS Option *User's Guide for Transaction Router Services*
	Mainframe Connect DirectConnect for z/OS Option *User's Guide for DB2 Access Services* (for use with MainframeConnect for DB2 UDB)

Glossary

This glossary includes terms and definitions either used or paraphrased from the following sources:

- The *IBM Dictionary of Computing*.
- The *American National Dictionary for Information Systems*.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). We also used definitions from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1.

In this glossary, we use the following references:

- Contrast with refers to a term that has an opposite or different meaning.
- Compare with refers to a term that has a similar meaning.
- See also refers to terms that have a related meaning.

Words with these references are highlighted.

access service The named set of properties, used with a DirectConnect access service library, to which clients connect. Each DirectConnect Server can have multiple services.

Adaptive Server Enterprise The server in the Sybase Client-Server architecture. It manages multiple databases and multiple users, tracks the actual location of data on disks, maintains mapping of logical data description to physical data storage, and maintains data and procedure caches in memory.

address A character or group of characters that identifies a register, a particular part of storage, or some other data source or destination.

addressing mode A program attribute that refers to the address length a program is prepared to handle upon entry. In MVS/370, addresses can be 24 bits in length.

advanced program-to-program communication (APPC) Hardware and software that characterize the LU 6.2 architecture and its various implementations in products. See also **logical unit 6.2**.

API See **application program interface.**

APPC See **advanced program-to-program communication.**

APPC communications link Hardware and software configured to enable a remote transaction program to establish an APPC conversation with a partner transaction program in an SNA network. See also **Systems Network Architecture**.

application The use to which an information processing system is put; for example, a payroll application, an airline reservation application, a network application.

application program interface (API) A functional interface, supplied by an operating system or other licensed program, that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

assembler language A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

ASE/CIS Adaptive Server Enterprise/ Component Integration Services (formerly OmniConnect).

authority The right to access objects, resources, or functions.

batch A group of records or data processing jobs brought together for processing or transmission.

bridge In the connection of local loops, channels, or rings, the equipment and techniques used to match circuits and to facilitate accurate data transmission. A bridge connects networks or systems of the same or similar architectures, whereas a gateway connects networks or systems of different architectures. Contrast with gateway, router.

buffer A portion of storage used to hold input or output data temporarily.

bulk copy transfer A transfer method in which multiple rows of data are inserted into a table in the target database. See also **transfer**. Contrast with **destination-template transfer**.

character string A sequence of consecutive characters that are used as a value.

CICS See **Customer Information Control System**.

client — In client/server systems, the part of the system that sends requests to servers and processes the results of those requests. See also **client/server**. Contrast with **server**.

client application — Software that is responsible for the user interface, including menus, data entry screens, and report formats or an application that sends requests to another application that acts as a server. See also **client/server**.

client/server — An architecture in which the client is an application that handles the user interface and local data manipulation functions, while the server provides data processing access and management for multiple clients. See also **client application**.

Client Services Application (CSA) — A user-written CICS program, initiated on the host, that uses a Sybase API to invoke MainframeConnect for DB2 UDB as a client to Open ClientConnect or to Adaptive Server Enterprise. See also **application program interface**, **Client Services for CICS**.

Client Services for CICS — A host API, supplied by Sybase ICD, that invokes Open ClientConnect as a client to an access service for DB2 or Adaptive Server Enterprise. See also **application program interface**, **Customer Information Control System**, **Client Services Application**, **Open ServerConnect**.

COBOL (common business-oriented language) — A high-level programming language, based on English, that is used primarily for business applications.

command — An order for an action to take place.

command level — An operation performed for a particular command in a program.

commit — An instruction to a database to make permanent all changes made to one or more database files since the last commit or rollback operation, and to make the changed records available to other users. Contrast with **rollback**.

common business-oriented language — See **COBOL**.

compile — To translate all or part of a program that is expressed in a high-level language into a computer program that is expressed in an intermediate language, an assembly language, or a machine language.

concatenation — To append one term to another to make a combined term. For example, aaaabbb is a concatenation of aaaa and bbb.

connectivity — The capability to attach a variety of functional units without modifying them.

conversation — a) A dialog between a user and an interactive data processing system.

	b) Within the context of APPC, an exchange of information or a sequence of messages sent between two transaction programs. Conversations take place between two LUs over an established session. Also, a sequence of messages sent between two applications (for instance, client application and Adaptive Server Enterprise).
CSA	See **Client Services Application.**
Customer Information Control System (CICS)	An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.
database management system (DBMS)	A computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The software for using a database can be part of the database management system, or it can be a stand-alone database system.
Database 2 (DB2)	An IBM relational database management system.
data definition statement (DD statement)	A job control statement describing a data set associated with a specific job step. See **job control language**.
data record	A collection of items of information from the standpoint of its use in an application, as the user supplies it. The data record is stored physically separate from its associated control information in a control interval.
data source	A collection of data, such as a database.
data structure	The syntactic structure of symbolic expressions and their storage allocation characteristics. A data structure can be either program described or externally described.
data transfer	The movement, or copying, of data from one location and the storage of the data at another location.
datatype	In programming languages, a set of values together with a set of permitted operations.
DBMS	See **database management system**.
DB2	See **Database 2**.
DD statement	See **data definition statement.**
debug	To detect, locate, and eliminate errors in computer programs.
declaration	In a programming language, a meaningful expression that affects the interpretation of other expressions in that language.

destination-template transfer — A transfer method in which source data is briefly put into a template where the user can specify that some action be performed on it before execution against a target database. See also **transfer**. Contrast with **bulk copy transfer**.

DirectConnect — A Sybase ICD Open Server application that provides access management for non-Sybase databases, copy management (transfer), and remote systems management. Each DirectConnect consists of a server and one or more service libraries to provide access to a specific data source. DirectConnect replaces the products "MDI Database Gateway" and "OmniSQL Access Module."

DirectConnect for OS/390 — A Sybase ICD LAN-based solution that communicates with mainframe host components. It incorporates the functionality of the MDI Database Gateway and the Sybase Net-Library and includes LU 6.2 and TCP/IP support.

DirectConnect Manager — A Java application from Sybase that can be used in Windows and UNIX environments. It provides remote management capabilities for DirectConnect products, including starting, stopping, creating, and copying services.

DLL — See **dynamic link library**.

download — To transfer data from a computer to a connected device, such as a workstation or microcomputer.

dump — To record, at a particular moment, the contents of all or part of one storage device in another storage device. Dumping is usually for the purpose of debugging.

dynamic link library (DLL) — A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a dynamic link library can be shared by several applications simultaneously.

dynamic SQL — Pertaining to the preparation and processing of SQL source statements within a program while the program runs. The SQL source statements are contained in host-language variables rather than being coded directly into the application program. The SQL statement can change several times while the program runs. Contrast with **static SQL**.

end-of-file — A coded character recorded on a data medium to indicate the end of the medium or the end of data.

end user — A person who connects to DirectConnect using an application in order to access databases and perform transfers. See also **transfer**.

Enterprise Mainframe Access Products — Sybase products that enable client applications to communicate with mainframes in a client/server environment. See **client/server**.

error message A message that a program issues, usually to the client application, when it detects an error condition.

execute To carry out an instruction.

field The smallest identifiable part of a record.

file A collection of related data that is stored and retrieved by an assigned name.

format In programming languages, a language construct that specifies the representation, in character form, of data objects in a file.

gateway Connectivity software that allows two or more computer systems with different network architectures to communicate.

halfword A contiguous sequence of bits or characters that constitutes half a computer word and can be addressed as a unit.

host The mainframe or other machine on which a database, an application, or a program resides.

index A set of pointers (that are logically ordered by the values of a key) that provide faster access to data and can enforce uniqueness on the rows in a table.

interface Hardware, software, or both, that links systems, programs, or devices.

JCL See **job control language.**

JCL statement A statement in a job that is used in identifying the job or describing its requirements to the operating system.

job control language (JCL) In MVS, a control language used to identify a job to an operating system and to describe the job's requirements.

LAN See **local area network**.

library A named area on disk that can contain programs and related information (not files). A library consists of different sections, called library members.

linkage In computer security, combining data or information from one information system with data or information from another system with the intention to derive additional information; for example, the combination of computer files from two or more sources.

linkage editor A computer program for creating load modules from one or more object modules or creating load modules by resolving cross references among the modules and, if necessary, adjusting addresses.

link-edit — To create a loadable computer program by means of a linkage editor. See also **linkage editor**.

load module — All or part of a computer program in a form suitable for loading into main storage for execution. A load module is usually the output of a linkage editor.

local area network (LAN) — A computer network located on the user's premises and covering a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary can be subject to some form of regulation.

logical unit (LU) — A type of network-accessible unit that enables end users to gain access to network resources and communicate with each other. See also **end user**.

logical unit 6.2 (LU 6.2) — A type of logical unit that supports general communication between programs in a distributed processing environment. See also **advanced program-to-program communication**.

LU — See **logical unit.**

LU 6.2 — See **logical unit 6.2**.

macro — An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that can also specify values for parameters in the replaced instructions.

MainframeConnect for DB2 UDB — A Sybase ICD mainframe solution that provides dynamic access to DB2 data. It replaces the OmniSQL Access Module for DB2 and the functionality in the MDI Access Server. See also **Database 2**, **Multiple Virtual Storage**, **Customer Information Control System**.

mask — A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

message number — The number that uniquely identifies an error message.

mode — A set of properties that defines the characteristics of a session between two LUs. Mode names must match exactly.

Multiple Virtual Storage (MVS) — An IBM operating system that runs on most System/370 and System/390 mainframes. It supports 24-bit addressing up to 16 megabytes.

MVS — See **Multiple Virtual Storage**.

network — A configuration of data processing devices and software connected for information exchange.

null A pointer that does not point to a data object.

object code Output from a compiler or assembler that is also itself executable machine code or is suitable for processing to produce executable machine code. Contrast with source code.

Open ClientConnect A Sybase ICD product that provides capability for the mainframe to act as a client to LAN-based resources. See **client**.

Open ServerConnect A Sybase ICD product that provides capability for programmatic access to mainframe data. It allows workstation-based clients to execute customer-written mainframe transactions remotely. Mainframe programmers use Open ServerConnect's Gateway-Library to accept remote requests and return results.

parameter A variable that is given a constant value for a specified application and that can denote the application.

password A value used in authentication or a value used to establish membership in a group of people having specific privileges.

pipe To direct data so that the output from one process becomes the input to another process. The standard output of one command can be connected to the standard input of another with the pipe operator (|). Two commands connected in this way constitute a pipeline.

platform The operating system environment in which a program runs.

PL/I See Programming Language/I.

Programming Language/I A programming language designed for use in a wide range of commercial and scientific computer applications.

pointer A data element that indicates the location of another data element.

precompile To process programs containing SQL statements before they are compiled. SQL statements are replaced with statements that will be recognized by the host language compiler. The output from this precompile includes source code that can be submitted to the compiler and used in the bind process.

program A sequence of instructions that a computer can interpret and execute.

program library An organized collection of computer programs, or parts of computer programs, and possibly information pertaining to their use. A program library is often called according to the characteristic of its elements; for example, a procedure library, a source program library.

programming language — An artificial language for expressing computer programs.

RDO — See **Resource Definition Online**.

record — A set of one or more related data items grouped for processing.

relational database — A database in which data is viewed as being stored in tables consisting of columns (data items) and rows (units of information). Data from different tables can be combined to form new data relationships.

remote stored procedure (RSP) — A customer-written CICS program that resides on the mainframe and communicates with MainframeConnect for DB2 UDB. See also **Customer Information Control System**. Contrast with **Client Services Application**.

Resource Definition Online (RDO) — A CICS interactive facility to create and modify system resources.

rollback — An instruction to a database to back out of the changes requested in a unit of work. Contrast with **commit**.

router — An attaching device that connects two LAN segments, which use similar or different architectures, at the reference model network layer. Contrast with **bridge**.

RSP — See **remote stored procedure**.

run time — A synonym for execution time.

server — A functional unit that provides shared services to workstations over a network. Contrast with **client**. See **client/server**.

service — A functionality available to DirectConnect applications. It is the pairing of a service library and a set of specific configuration properties.

SNA — See **Systems Network Architecture**.

source code — The input to a compiler or assembler, written in a source language. Contrast with object code.

source language — A language from which statements are translated.

SPAREA (Stored Procedure Communication Area) — An area in which a RSP exchanges information with Open ServerConnect.

SQL — See **structured query language**.

SQLCA (SQL communication area) A set of variables that are used by SQL to provide an application program with information about the processing SQL statements within the program.

SQL communication area See **SQLCA**.

SQLDA (SQL descriptor area) A set of variables used in the processing of certain SQL statements. The SQLDA is intended for dynamic SQL programs.

SQL descriptor area See **SQLDA**.

staging The movement of data from an off-line or low-priority device back to an online or higher-priority device, usually on demand of the system or on request of a user.

statement A basic unit of SQL, which is a single SQL operation, such as select, update, or delete.

static SQL SQL statements that are embedded within a program and are prepared during the program preparation process before the program runs. After being prepared, the statement itself does not change, although values of host variables specified by the statement can change. Contrast with **dynamic SQL**.

Stored Procedure Communication Area See **SPAREA.**

structured query language (SQL) An IBM industry-standard language for processing data in a relational database.

stub A program module that transfers remote procedure calls and responses between a client and a server. See **client**, **server**.

syntax The rules for how to construct a statement.

Systems Administrator A user authorized to handle Open ClientConnect system administration, including creating user accounts, assigning permissions, and creating new databases.

Systems Network Architecture (SNA) An IBM proprietary plan for the logical structure, formats, protocols, and operational sequences for transmitting information units through networks and controlling network configuration and operation. See also **advanced program-to-program communication**.

target A system, program, or device that interprets, rejects or satisfies, and replies to requests received from a source.

temporary storage In computer programming, storage locations reserved for intermediate results.

trace The process of recording the sequence in which the statements in a program are executed and, optionally, the values of the program variables used in the statements.

transaction An exchange between a program on a local system and a program on a remote system that accomplishes a particular action or result.

transfer A DirectConnect feature that allows users to move data or copies of data from one database to another. See also **bulk copy transfer, destination-template transfer**.

troubleshoot To detect, locate, and eliminate errors in computer programs or faults in hardware.

user ID User identification.The ID number by which a user is known in a specific database or system.

user interface Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

Virtual Telecommunications Access Method (VTAM) A set of programs that maintain control of the communication between terminals and application programs running under DOS/VS, OS/VS1, OS/VS2, and MVS operating systems.

VTAM See **Virtual Telecommunications Access Method**.

Windows New Technology (Windows NT) A multi-tasking operating system from Microsoft Corporation.

Windows NT See **Windows New Technology**.

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W