SYBASE®

XA Interface Integration Guide for CICS, Encina, and TUXEDO

Adaptive Server Enterprise

12.5

DOCUMENT ID: 36123-01-1250-01

LAST REVISED: June 2001

Copyright © 1989-2001 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase database management software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, the Sybase logo, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, AnswerBase, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-FORMS, APT-Translator, APT-Library, Backup Server, ClearConnect, Client-Library, Client Services, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, E-Anywhere, E-Whatever, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, EWA, Financial Fusion, Financial Fusion Server, Gateway Manager, ImpactNow, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InternetBuilder, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, MainframeConnect, Maintenance Express, MAP, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, MySupport, Net-Gateway, Net-Library, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RW-DisplayLib, RW-Library, S-Designor, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, STEP, SupportNow, Sybase Central, Sybase Client/Server Interfaces, Sybase Financial Server, Sybase Gateways, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, Transact-SQL, Translation Toolkit, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Viewer, Visual Components, VisualSpeller, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server and XP Server are trademarks of Sybase, Inc. 3/01

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Contents

About This Book ..... **v**

CHAPTER 1 **Introduction** **1**
Requirements 2

CHAPTER 2 **The Sybase XA Environment** **3**
Definitions 3
Overview of the X/Open DTP model 4
Components of the model 5
How the components communicate 6
How the components interact 6
Recovery 8
The Sybase XA environment 9
Components of the Sybase XA environment 10
Connections in the Sybase XA environment 11
Identifying connections via LRMs 11
Establishing connections 13
Distributing work across LRMs 14

CHAPTER 3 **Configuring the XA Environment** **17**
Configuring Adaptive Server 17
Open string parameters for DTM XA Interface 18
Open string parameters 18
dtm_tm_role required for username 19
Log file and trace flag parameters 19
xa_open() function behavior 20
XA configuration file for DTM XA Interface 20
Environment variable for specifying configuration file 20
[all] section for defining common LRM parameters 21
Editing the XA configuration file 22
Additional capabilities, properties, and options 24
Using the DTM XA Interface with CICS 26
Building the switch-load file 26

Adding a Sybase stanza to the CICS region XAD definition ... 30
Using the DTM XA Interface with Encina ... 31
Assigning an open string with monadmin create rm ... 31
Initializing LRMs with mon_RegisterRmi ... 32
Linking applications with DTM XA Interface libraries ... 32
Establishing connections ... 33
Using the DTM XA Interface with TUXEDO ... 33
Linking ... 35
Setting up the UBBCONFIG file ... 36
Creating the TUXEDO configuration file ... 37
Building the TMS ... 38
Build COBOL runtime environment ... 38

CHAPTER 4 Application Programming Guidelines ... 41
X/Open DTP versus traditional Sybase transaction processing ... 41
Transaction and connection management ... 42
Transaction management ... 42
Connection management ... 43
The current connection ... 44
Nontransactional connections ... 45
Deallocate cursor function with Client-Library ... 45
Dynamic SQL ... 46
Getting a Client-Library connection handle ... 46
Multiple-thread environment issues ... 49
Caveats of thread use ... 50
Embedded SQL thread-safe code ... 51
Tightly coupled transactions ... 51
Linking with CT Library ... 53
Sample embedded SQL COBOL fragment ... 53
Sample embedded SQL C fragment ... 55

Index ... 59

About This Book

Audience

This guide serves as a reference manual for:

- System administrators setting up a distributed transaction processing (DTP) environment that includes one or more Adaptive Servers with distributed transaction management features, accessed by transactions from within a CICS, Encina, or TUXEDO transaction manager (TM) system.
- Application programmers using Embedded SQL™ or Client-Library™ to access data on one or more Adaptive Servers.

This manual assumes the reader is familiar with:

- The TM operating environment
- Embedded SQL
- Open Client™ Client-Library
- Adaptive Server® administration

How to use this book

Use this guide to help configure your environment and code your application to access data stored on one or more Adaptive Servers from within a CICS, Encina, or TUXEDO TM.

Chapter 1, “Introduction” summarizes the steps necessary to fully integrate the DTM XA Interface into your environment.

Chapter 2, “The Sybase XA Environment” provides background information designed to help you place the Sybase XA environment into the larger context of distributed transaction processing and transaction management. It reviews the X/Open DTP model of distributed transaction processing and fits the Sybase DTM XA Interface into this model. In addition, it describes how the individual components of the Sybase® XA environment work together to allow your application to access Adaptive Server data from a TM.

Chapter 3, “Configuring the XA Environment” gives instructions for configuring your environment to fully integrate your application, Sybase DTM XA Interface, one or more Adaptive Servers, and your TM software.

Chapter 4, "Application Programming Guidelines" explains how to make your Embedded SQL or Client-Library application conform to certain coding constraints that the Sybase XA environment imposes.

Related documents

The *Installation Guide* for your platform explains how to install Adaptive Server and the DTM XA Interface. It also describes how to install licenses for Adaptive Server features such as distributed transaction management.

To use this manual, you should also be familiar with the information described in the following manuals:

- *Using Adaptive Server Distributed Transaction Management Features*
- *X/Open CAE Specification (December 1991) Distributed Transaction Processing: The XA Specification*
- *Open Client Embedded SQL/COBOL Programmer's Guide or Open Client Embedded SQL/C Programmer's Guide*
- *Open Client Embedded SQL Reference Manual*
- *Open Client-Library/C Reference Manual*
- *Open Client-Library/C Programmer's Guide*
- *System Administration Guide*
- Your CICS, Encina, or TUXEDO TM documentation set

Other sources of information

Use the Sybase Technical Library CD and the Technical Library Product Manuals Web site to learn more about your product:

- Technical Library CD contains product manuals and is included with your software. The DynaText browser (downloadable from Product Manuals at http://www.sybase.com/detail/1,3693,1010661,00.html) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.

- Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to the Technical Documents Web site (formerly known as Tech Info Library), the Solved Cases page, and Sybase/Powersoft newsgroups.

  To access the Technical Library 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.

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

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **For the latest information on EBFs and Updates**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select EBFs/Updates. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).
3. Specify a time frame and click Go.
4. Select a product.
5. Click an EBF/Update title to display the report.

❖ **To create 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.

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

The DTM XA Interface is Sybase's implementation of the XA Interface standard, which is one element of the X/Open Distributed Transaction Processing (DTP) model. The X/Open DTP model provides an industry standard for developing distributed transaction processing applications.

Use the XA Interface to access data stored on Adaptive Servers from within a CICS, Encina, or TUXEDO TM. To use native Adaptive Server distributed transaction management (DTM) features with or without a TM, see *Using Adaptive Server Distributed Transaction Management Features*.

The Microsoft Transaction Server (MTS) also uses the XA Interface for transactional components. MTS XA configuration is handled through the Sybase ODBC driver setup. See the ODBC driver documentation for details.

Other TMs, such as IBM MQ Services, also use the XA Interface. See the transaction manager documentation for details on how to configure the TM.

To enable a TM transaction to access data stored on Adaptive Server, you must:

1 Install Adaptive Server, and the distributed transaction management feature. Software installation and feature licenses are described in the Adaptive Server *Installation Guide* for your platform.

Note Distributed transaction management is a separately-licensed Adaptive Server feature. You must purchase and install a valid license for DTM before it can be used.

---

2 Install Sybase Open Client on all client machines. The DTM XA Interface is included with Open Client.

3 Start Adaptive Server with support for the distributed transaction management feature. See *Using Adaptive Server Distributed Transaction Management Features* for information.

4 Configure the TM software to run with an Embedded SQL or Client-Library application and Adaptive Server, as described in Chapter 3, "Configuring the XA Environment."

5 Make the Embedded SQL or Client-Library application conform to certain coding constraints, as described in Chapter 4, "Application Programming Guidelines."

6 Start the CICS, Encina, or TUXEDO TM.

Note To administer global recovery manually in the Sybase XA environment, you must invoke XA-specific dbcc commands, as described in *Using Adaptive Server Distributed Transaction Management Features*.

Requirements

XA Interface for Adaptive Server version 12.5 is compatible with:

- Open Client 12.5
- Embedded SQL 12.0 or later
- Adaptive Server 12.0 or later
- CICS/6000 2.1.1.6
- Encina 2.5/TX Series 4.2
- TUXEDO 6.4 (6.3/6.4 on IBM platforms)

CHAPTER 2

The Sybase XA Environment

This chapter describes the X/Open DTP model, and shows how the components of the Sybase XA environment—including the DTM XA Interface, your application program, and Adaptive Server, among others—fit into that model. It also discusses how connections are established and managed in the Sybase XA environment.

Topic	Page
Definitions	3
Overview of the X/Open DTP model	4
The Sybase XA environment	9
Connections in the Sybase XA environment	11

Definitions

The X/Open DTP model assumes an understanding of certain terms.

- **transaction** – a whole unit of work consisting of one or more computational tasks. Most often, a transaction's tasks manipulate shared resources.
- **committed transaction** – a completed transaction whose changes to any shared resources are permanent.
- **rolled-back transaction** – a complete transaction whose changes to any shared resources are nullified.
- **ACID test** – the test of a true transaction; to pass, the transaction must exhibit the following properties:
    - **Atomicity** – all or none of the results of the transaction take effect.
    - **Consistency** – if a transaction is rolled back, all resources that the transaction affected return to the state they were in prior to the transaction's execution.

  - **Isolation** – a transaction's results are visible only to that transaction until the transaction commits.
  - **Durability** – permanent resource changes resulting from commitment survive subsequent system failures.
- **transaction processing** – a system of coordinating the transactions that multiple users perform on shared, centralized resources.
- **distributed transaction processing** – a transaction processing model in which the shared resources are located at distinct physical sites on a computer network.
- **local transaction** – a transaction that affects data in a single database and whose tasks a single resource manager performs. See "Overview of the X/Open DTP model" on page 4 for a definition of resource managers.
- **global transaction** – a transaction that spans more than one database and multiple resource managers.
- **transaction branch** – a portion of the work that makes up a global transaction.
- **transaction identifier** – an identifier that a TM assigns to a transaction. The transaction monitor uses the transaction identifier to coordinate all activity related to a global transaction. The resource manager uses the global identifier to match the recoverable tasks it performed for the transaction.
- **recovery** – the process of bringing a transaction processing system into a consistent state after a failure. Specifically, this means resolving transactions left in a noncommitted state.

Overview of the X/Open DTP model

The X/Open DTP model is a model for software architecture that allows multiple application programs to share resources provided by multiple resource managers, and allows their work to be coordinated into global transactions.

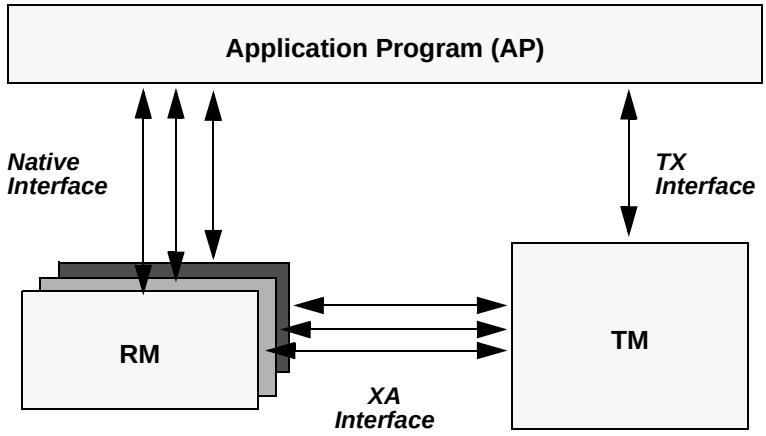
The X/Open DTP model identifies the key entities in a distributed transaction processing environment and standardizes their roles and interactions. The entities are:

- The transaction processing monitor (TM)

- The resource manager (RM)
- The application program (AP)

This section discusses the X/Open DTP functional model, including its major components and their interfaces. Figure 2-1 shows the X/Open DTP model.

Figure 2-1: A conceptual view of the X/Open DTP model



These components communicate through the native, XA, and TX Interfaces as described in "How the components communicate" on page 6.

Components of the model

The X/Open DTP functional model consists of the following components:

- The application program (AP)
- The resource manager (RM)
- The transaction processing monitor (TM)

The AP contains the code written to accomplish a particular transaction or portion thereof. As such, it designates the beginning and end of global transactions.

The RM provides access to shared resources. Database servers, file servers, and print servers are examples of RMs. In a typical X/Open DTP environment, a single AP communicates with more than one RM. In the Sybase XA environment, the RM is an Adaptive Server database.

The TM coordinates the communication between all parties participating in the transaction. The TM assures that the work done by the AP is contained in a global transaction, which commits or aborts atomically.

Specifically, the TM's tasks include:

- Assigning global identifiers to transactions.
- Monitoring the progress of global transactions.
- Coordinating the flow of transaction information between the APs and the RMs.
- Managing the transaction commitment protocol and failure recovery. For details, see "How the components interact" on page 6.

How the components communicate

The AP, the RM, and the TM communicate via three distinct interfaces: native, TX, and XA.

The native interface is the medium by which the AP makes requests directly to the RM. This interface is RM specific. In the Sybase XA environment, the native interface is either Embedded SQL or Client-Library.

The TX Interface is the medium between the AP and the TM. The AP uses TX calls to delineate transaction boundaries. In other words, the AP requests that the TM start and commit or roll back global transactions, via the TX Interface. This interface is TM specific.

The XA Interface is the medium between the RM and the TM. The DTM XA Interface is Sybase's version of the interface for Adaptive Server. Using XA calls, the TM tells the RM when transactions start, commit, and roll back. The TM also handles recovery.

How the components interact

The components work together to process transactions from initiation through completion.

The AP delimits transaction boundaries. An AP informs the TM, via TX calls, that a global transaction is beginning. The TM then communicates with all available RMs, via XA calls, to associate a single transaction identifier with any work the RMs will do on behalf of the AP within the bounds of the global transaction.

When the AP requests that the TM commit the global transaction, the TM and the RMs use the two-phase commit protocol to guarantee transaction atomicity.

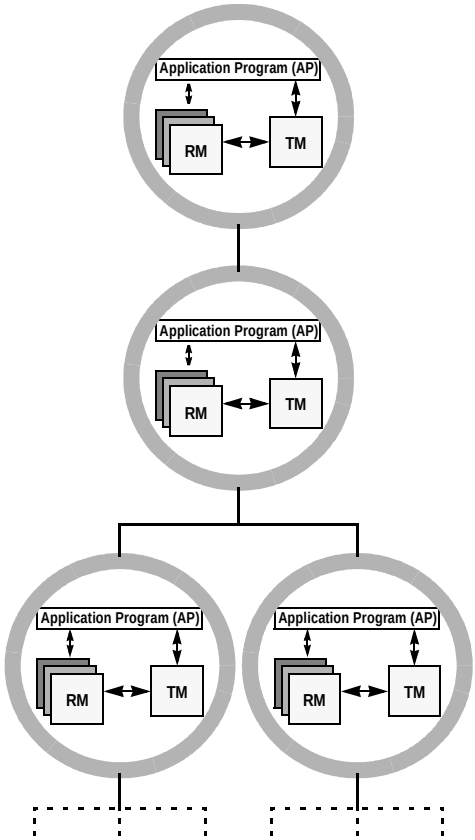
Transaction completion takes place in two phases—the prepare phase and the commit phase. For a detailed description of the two-phase commit protocol, see the *Open Client DB-Library/C Reference Manual*.

In the prepare phase, the TM requests each RM to prepare to commit its portion of the global transaction. This portion is known as a **transaction branch**.

In the commit phase, the TM instructs the RMs to commit or abort their branches of the transaction. If all RMs report back that they have prepared their respective transaction branches, the TM commits the entire transaction. If any RM reports that it was unprepared or fails to respond, the TM rolls back the entire transaction.

Figure 2-2 shows a typical transaction branch structure.

Figure 2-2: Transaction branches



Recovery

The TM is responsible for managing global recovery. In certain situations, an administrator may decide to complete its transaction branch independently of the TM. When this occurs, the administrator's decision is called a *heuristic decision.*

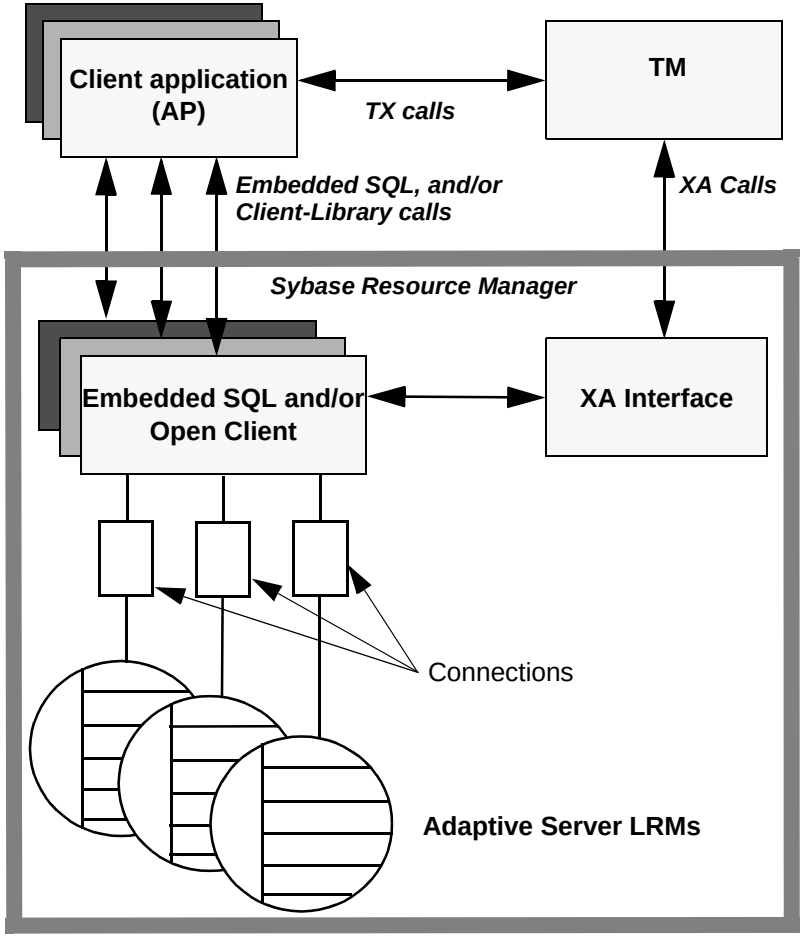
The heuristic decision may be in conflict with the TM's decision. For example, the administrator may commit a transaction branch and the TM may request to abort it.

Such a conflict requires manual intervention from the System Administrator. For a discussion of heuristic decisions in the Sybase XA environment, see *Using Adaptive Server Distributed Transaction Management Features*.

The Sybase XA environment

The DTM XA Interface relies on Sybase's transaction processing model to implement X/Open's DTP model. Adaptive Server is used as an RM, as shown in Figure 2-3.

Figure 2-3: The Sybase XA DTP model



Components of the Sybase XA environment

The Sybase XA environment consists of:

- The Sybase DTM XA Interface. This is Sybase's implementation of the XA Interface for Adaptive Server, described in "How the components communicate" on page 6.
- The Open Client libraries. Client-Library calls can be part of the native interface between your application and the resource manager.

- Embedded SQL/C and Embedded SQL/COBOL. Embedded SQL calls can be part of the native interface between your application and the resource manager.
- One or more Adaptive Servers. These play the role of RMs.
- The XA configuration file. This file contains entries that define client/server connections for use with XA.
- A set of XA-specific dbcc commands. System Administrators use these to manage heuristic transactions.
- TM-specific configuration files and commands.

Chapter 3, “Configuring the XA Environment” explains how to configure these components so that transactions can use the DTM XA Interface to access data stored on Adaptive Server.

Connections in the Sybase XA environment

The X/Open DTP model has no notion of connections, yet connections are central to the Sybase client/server architecture. The Sybase XA environment must resolve this discrepancy.

To this end, the Sybase XA environment introduces the notion of a logical resource manager (LRM).

Identifying connections via LRMs

Each instance of the Sybase RM appears to the TM as one or more LRMs.

An LRM associates a symbolic name with a client/server connection. An AP uses the names to identify the specific physical connection to one or more Adaptive Servers. The TM uses the names to open connections on behalf of the AP.

Where is the connection information stored?

The following components of the Sybase XA environment contain information about LRMs. The System Administrator configures these files before starting up the TM. For information on the full configuration process, see Chapter 3, "Configuring the XA Environment."

The Sybase XA configuration file

The Sybase XA configuration file contains one entry per LRM. The entry associates the LRM with a physical Adaptive Server name, and assigns pre-connection Client-Library capabilities and properties to the LRM. For details on the XA configuration file, see "XA configuration file for DTM XA Interface" on page 20.

The CICS XA product definition (XAD)

The CICS XAD contains one stanza per LRM. The stanza assigns each LRM a user name and password in the form of an open string. The user name and password enable the Sybase XA environment to control a particular connection's access to Adaptive Server resources. For details on the CICS XAD file, see "Adding a Sybase stanza to the CICS region XAD definition" on page 30.

The Encina *monadmin create rm* command

The monadmin create rm command assigns each LRM a user name and password in the form of an *open string*. The user name and password allow the Sybase XA environment to control a particular connection's access to Adaptive Server resources. For details on the Encina monadmin command, see "Assigning an open string with monadmin create rm" on page 31. Your current version of Encina may have additional commands for specifying RMs.

Note You can use the Encina enconsole interactive command instead of the shell monadmin command.

For detailed information, see the *Encina Monitor System Administrator's Guide and Reference*.

TUXEDO's UBBCONFIG file

In addition to modifying the Sybase configuration files, integrating TUXEDO requires customizing the TUXEDO configuration file, *UBBCONFIG*. The *open string* is the only portion of the *UBBCONFIG* file that requires modification. It includes the user name and password, which allow XA-Server to control a connection's access to Adaptive Server resources. See "Setting up the UBBCONFIG file" on page 36 for details.

Establishing connections

The TM, together with the XA Interface, establishes connections between applications and RMs in several steps.

❖ **Establishing connections in a CICS environment**

1 When the CICS region starts up, it issues an XA open call to each LRM configured in an XAD, using the information contained in each open string.

2 The CICS region passes to the XA Interface library the open string associated with each stanza. The open string contains the LRM name, the user name, and the password.

3 The XA Interface looks up the LRM name in the Sybase XA configuration file and matches it to an actual RM name, that is, an actual physical Adaptive Server. The RM name matches an entry in the Adaptive Server *interfaces* file.

4 The XA Interface establishes one connection to an Adaptive Server for each LRM entry. The XA Interface confers on any connection the pre-connection properties and capabilities configured for the LRM.

❖ **Establishing connections in an Encina environment**

1 An application issues a mon_RegisterRmi function, thereby requesting use of an LRM.

2 Using information contained in an open string, the TM issues an XA open call to the LRM (configured in the monadmin create rm command) whose name matches that issued in step 1, above.

3 The TM passes the open string associated with each monadmin create rm command to the XA Interface. The open string contains the LRM name.

4 The XA Interface looks up the LRM name in the Sybase XA configuration file and matches it to an actual RM name—that is, to an actual physical Adaptive Server. The RM name matches an entry in the Adaptive Server *interfaces* file.

5 The XA Interface establishes one logical connection to an Adaptive Server for each LRM entry. The XA Interface confers on any connection the pre-connection properties and capabilities configured for the LRM.

❖ **Establishing connections in a TUXEDO environment**

1 The application uses the LRM specified in the *UBBCONFIG* file to reference the logical connection for a branch of a global transaction. In using the LRM name, the application implicitly requests and establishes an LRM.

2 The transaction manager passes the appropriate open string to the XA Interface through the LRM whose name matches the one issued in step 1. The XA Interface uses the LRM name, the user name, and the password.

3 The XA Interface looks in the *xa_config* file to find an association between the LRM name and Adaptive Server. The Adaptive Server name matches an entry in the *interfaces* file where its network information is kept.

4 The XA Interface establishes one logical connection to an Adaptive Server for each LRM entry. The XA Interface confers on any connection the pre-connection properties and capabilities configured for the LRM.

Distributing work across LRMs

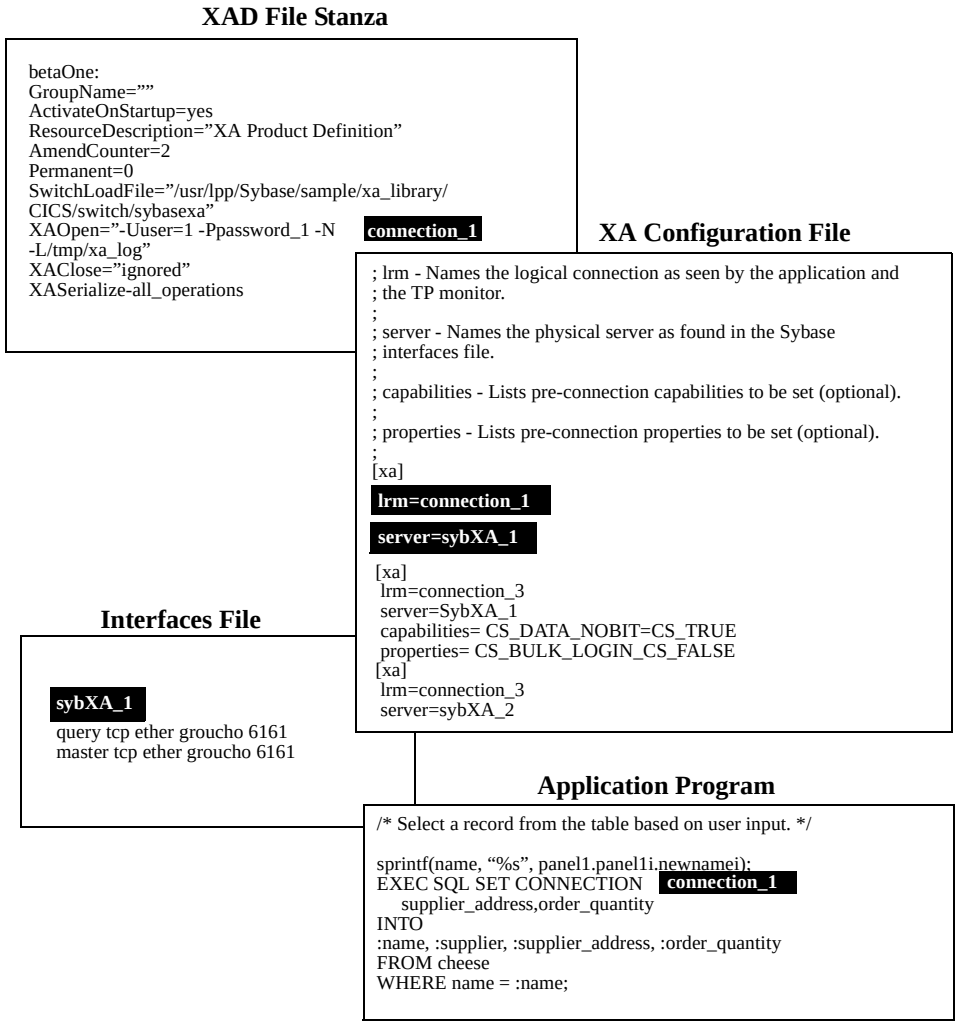
The System Administrator and the application programmer together must agree on the number and names of LRMs that their Sybase XA environment includes.

The System Administrator configures the TM and Sybase XA configuration files accordingly. The application programmer invokes a particular LRM name within the application code to send a portion of a global transaction across that connection. The TM coordinates this distribution.

You can configure the Sybase XA environment for more connections than are actually used. That is, the XA configuration file may contain inactive entries.

Figure 2-4 depicts the relationship between the Sybase XA configuration file, TM configuration file, application code, and Adaptive Server *interfaces* file for a CICS environment.

Figure 2-4: Relating components of the Sybase XA environment for CICS



CHAPTER 3

Configuring the XA Environment

This chapter describes how to configure the XA environment for use with CICS, Encina, and TUXEDO TMs.

Topic	Page
Configuring Adaptive Server	17
Open string parameters for DTM XA Interface	18
XA configuration file for DTM XA Interface	20
Using the DTM XA Interface with CICS	26
Using the DTM XA Interface with Encina	31
Using the DTM XA Interface with TUXEDO	33
Build COBOL runtime environment	38

You must link the DTM XA Interface library with your X/Open XA-compliant transaction manager to use Adaptive Server as a resource manager.

Note See the *README* file under the subdirectories of *$SYBASE/$SYBASE_OCS/sample* for detailed information about configuring the DTM XA Interface for your system.

Configuring Adaptive Server

To function in a Sybase XA environment, your Adaptive Server must be licensed and configured to use the distributed transaction management feature. See the Adaptive Server *Installation Guide* and *Using Adaptive Server Distributed Transaction Management Features* for more information.

If your Adaptive Server is licensed to use distributed transaction management, you can enable the feature using the enable dtm configuration parameter:

```
sp_configure 'enable dtm', 1
```

You must restart Adaptive Server for this parameter to take effect.

Open string parameters for DTM XA Interface

The X/Open XA specification allows each resource manager vendor to define an open string and a close string. The DTM XA Interface does not require or use the close string.

The DTM XA Interface uses the required and optional open string parameters described below.

Open string parameters

The format for parameters in the open string for the DTM XA Interface is:

-N*lrm_name* -U*username* -P*password* [-L*logfile_name*]
[-T*traceflags*] [-V11] [-O1] [-O-1]

Table 3-1 describes each component of the open string.

Table 3-1: Sybase X/Open XA open string parameters

Parameter	Meaning
lrm_name	The name of the LRM as defined in the XA configuration file.
username	The user name used to log in to Adaptive Server. See "*dtm_tm_role* required for *username*" below for more information.
password	The password accompanying the user name.
logfile_name	The fully qualified file name to which the XA Interface writes tracing information (optional). The XA Interface initializes the log file and trace flag settings with the initial xa_open() call. If no *logfile_name* is specified, then the DTM XA Interface logs information to a file named *syb_xa_log* in the current directory.
traceflags	Trace flags control the output that is written to the log file (optional). See "Parameters for [all] section of XA configuration file" in this document for a list of valid trace flags.
-V11	Specifies Open Client version 11 behavior for backward compatibility (optional).

Parameter	Meaning
-O1 or -O-1	Specifies an option for transaction operation. At this time only -O1 (tightly coupled transaction branches) is supported. -O1 sets the option. -O-1 clears the option. -O-1 is the default. **Warning!** See "Tightly coupled transactions" in chapter 4 of this document before setting -O1.

dtm_tm_role required for *username*

In the open string for resource managers, the specified *username* must have the dtm_tm_role in the corresponding Adaptive Server. The System Security Officer can assign this role using sp_role or the grant command. For example:

```
sp_role "grant", dtm_tm_role, user_name
```

Log file and trace flag parameters

With the DTM XA Interface to Adaptive Server, log file and trace flag parameters can be defined in the [all] section of the XA configuration file, rather than in the X/Open XA open string. See "[all] section for defining common LRM parameters" on page 21 for more information about the log file and trace flag components.

Labels for logfile entries

The DTM XA Interface marks each entry in the log file with a label indicating the severity or cause of the message. Table 3-2 describes each label.

Table 3-2: Logfile message labels

Label	Type of entry
Error	An error returned to the transaction manager
Fatal Error	A severe failure in the DTM XA Interface, with a possible application or transaction manager error
Message	Additional information about a previous error, or a description of the operational environment
Warning	A condition that may indicate problems with the transactional system

Label	Type of entry
Note	Information that does not indicate a problem, but may be useful if an error occurs
XA trace	Information logged as a result of the xa trace flag setting
RM trace	Information logged as a result of the xl trace flag setting
Connection trace	Information logged as a result of the xc trace flag setting
ASE I/F trace	Information logged as a result of the xs trace flag setting
Misc trace	Information logged as a result of the misc trace flag setting
Event trace	Information logged as a result of the event trace flag setting
Verbose trace	Information logged as a result of the v trace flag setting
Function trace	Information logged as a result of the cmn trace flag setting
Open Client trace	Information logged as a result of the ct trace flag setting

xa_open() function behavior

The X/Open XA function, xa_open(), initiates a single connection to Adaptive Server. The *username* and *password* defined in the open string must possess the dtm_tm_role in the server, as described under “dtm_tm_role required for username” on page 19.

XA configuration file for DTM XA Interface

The DTM XA Interface to Adaptive Server provides mechanisms for configuring the Open Client connections using the XA configuration file. Set all connection capabilities, properties, and options using the XA configuration file.

Environment variable for specifying configuration file

The DTM XA Interface uses the environment variable *XACONFIGFILE* to find the full path and file name of the XA configuration file. You can set this environment variable to specify different locations and names to use for configuration information as necessary.

For example, on UNIX platforms:

```
setenv XACONFIGFILE /usr/u/sybase/xaconfig1.txt
```

If *XACONFIGFILE* is not defined, or if it does not specify a valid configuration file, the DTM XA Interface looks for a file named *xa_config* in the following directories:

- *$SYBASE/$SYBASE_OCS/config*
- *$SYBASE/$SYBASE_OCS*
- *$SYBASE/config*
- *$SYBASE*

The DTM XA Interface uses the first *xa_config* file it finds.

[all] section for defining common LRM parameters

The DTM XA Interface uses the [all] section to define parameters that apply to all LRMs.

Certain parameters defined in the [all] section—log file and trace flag definitions—may also be defined in the open string for X/Open XA transaction managers.

Parameter definitions for [all] section

Entries for the [all] section in the XA configuration file are as follows:

```
[all]
logfile=logfile_name
traceflags=[xa | xl | xc | cm| event | misc | os | ct |
all]
[properties=name=value] [...]
```

Table 3-3 describes each component.

Table 3-3: Parameters for [all] section of XA configuration file

Parameter	Meaning
logfile_name	The fully qualified file name to which the DTM XA Interface writes tracing information. The DTM XA Interface initializes the log file and trace flag settings with the initial xa_open() call.

Parameter	Meaning
traceflags	The trace flags control the output that is written to the log file. Specify one or more of the following flags: all – all tracing. ct – the ct_debug option with the CS_DBG_ERROR flag (ct_debug functionality is available only from within the debug version of Client-Library). cmn – entry and exit point tracing of internal XA Interface functions. event – tracing of significant internal events. misc – tracing of activities and information for problem resolution. xa – entry and exit point tracing at the xa_* level. xc – entry and exit point tracing at the xc_* level. xl – entry and exit point tracing at the xl_* level. **Note** Tracing at the xc_*, xl_*, event, misc, and cmn levels is intended to be meaningful only to Sybase development. Specify these tracing levels only when instructed to do so by Sybase Technical Support.
properties	**Note** The following property must be set in the [all] stanza. You cannot set it in the [xa] stanza: CS_LOGIN_TIMEOUT=*timeout* You can define these optional properties in the [all] section of the XA configuration file: PROPERTIES=CS_DISABLE_POLL=[CS_TRUE | CS_FALSE] PROPERTIES=CS_EXTRA_INF=[CS_TRUE | CS_FALSE] PROPERTIES=CS_HIDDEN_KEYS=[CS_TRUE | CS_FALSE] PROPERTIES=CS_MAX_CONNECT=*number_of_connections* PROPERTIES=CS_NOINTERRUPT=[CS_TRUE | CS_FALSE] PROPERTIES=CS_TEXTLIMIT=*textlimit* PROPERTIES=CS_TIMEOUT=*timeout*

Editing the XA configuration file

You must customize the XA configuration file for the application environment. Use the text editor of your choice to open the XA configuration for editing. The sample contents of an XA configuration file are as follows:

```
; Comment line as first line of file REQUIRED!
;
; xa_config - sample xa_config file.
;
; Note that the Adaptive Server names may need
; to be customized for your environment.

; simprpc.ct sample application entry.

[all]
       logfile=logfile_name
       traceflags=traceflags
       properties=name=value [, name=value] [...]

 [xa]
         lrm=connection1
         server=sqlserver

 ; Rentapp sample xa_config entries.

 [xa]
         lrm=FLEET_CON
         server=fleetsrv

 [xa]
       lrm=RESERVE_CON
       server=rsrvsrv
```

Note The first line of the *xa_config* file MUST be a comment which is denoted by a semicolon (;) in the first character position.

For each additional LRM, create an entry with the following format. Keep the connection1 entry for installation verification.

```
[xa]
<tab> lrm=connection_name
<tab> server=adaptive_server_name
<tab> capabilities=name=value [, name=value] [...]
<tab> properties=name=value [, name=value] [...]
<tab> options=name=value [, name=value] [...]
```

The *connection_name* is the symbolic name for the connection between the application and SQL. The *adaptive_server_name* is the name of the Adaptive Server associated with the connection. *adaptive_server_name* must correspond to a server name defined in the *interfaces* file.

See “Additional capabilities, properties, and options” on page 24 for information a list of capabilities, properties, and options that can be used with the DTM XA Interface.

Additional capabilities, properties, and options

XA configuration file entries for capabilities, properties, and options have the following general format:

```
<tab> capabilities=name=value [, name=value] [...]
<tab> properties=name=value [, name=value] [...]
<tab> options=name=value [, name=value] [...]
```

Table 3-4, Table 3-5, and Table 3-6 list the names for capabilities, properties, and options that can be defined in the XA configuration file for the DTM XA Interface. Unless otherwise specified in these tables, the valid values for each capability, property, or option are CS_TRUE or CS_FALSE.

Note All names and values for these capabilities, properties, and options correspond to CS-Library *keywords*. See the *Open Client Client-Library/C Reference Manual* for specific descriptions.

Table 3-4: XA Interface valid capabilities

Capabilities	
CS_CON_NOINBAND	CS_DATA_NOINT2
CS_CON_NOOOB	CS_DATA_NOINT4
CS_DATA_NOBIN	CS_DATA_NOINT8
CS_DATA_NOVBIN	CS_DATA_NOINTN
CS_DATA_NOLBIN	CS_DATA_NOMNY4
CS_DATA_NOBIT	CS_DATA_NOMNY8
CS_DATA_NOBOUNDARY	CS_DATA_NOMONEYN
CS_DATA_NOCHAR	CS_DATA_NONUM
CS_DATA_NOVCHAR	CS_DATA_NOSENSITIVITY
CS_DATA_NOLCHAR	CS_DATA_NOTEXT
CS_DATA_NODATE4	CS_PROTO_NOBULK
CS_DATA_NODATE8	CS_PROTO_NOTEXT
CS_DATA_NODATETIMEN	CS_RES_NOEED
CS_DATA_NODEC	CS_RES_NOMSG
CS_DATA_NOFLT4	CS_RES_NOPARAM

Capabilities	
CS_DATA_NOFLT8	CS_RES_NOTDSDEBUG
CS_DATA_NOIMAGE	CS_RES_NOSTRIPBLANKS
CS_DATA_NOINT1	

Table 3-5: XA Interface valid properties

Properties	
CS_ASYNC_NOTIFS	CS_SEC_NEGOTIATE
CS_DIAG_TIMEOUT	CS_TDS_VERSION= [CS_TDS_40 | CS_TDS_42 | CS_TDS_46 | CS_TDS_50]
CS_DISABLE_POLL	CS_TEXTLIMIT=*textlimit*
CS_HIDDEN_KEYS	CS_EXTRA_INF
CS_PACKETSIZE=*packetsize*	CS_MAX_CONNECT=*connections*
CS_SEC_APPDEFINED	CS_NOINTERRUPT
CS_SEC_CHALLENGE	CS_TIMEOUT=*timeout*
CS_SEC_ENCRYPTION	

Table 3-6: XA Interface valid options

Options	
CS_OPT_ANSINULL	CS_OPT_NOEXEC
CS_OPT_ANSIPERM	CS_OPT_PARSEONLY
CS_OPT_ARITHABORT	CS_OPT_QUOTED_IDENT
CS_OPT_ARITHIGNORE	CS_OPT_RESTREES
CS_OPT_DATEFIRST= [CS_OPT_SUNDAY | CS_OPT_MONDAY | CS_OPT_TUESDAY | CS_OPT_WEDNESDAY | CS_OPT_THURSDAY | CS_OPT_FRIDAY | CS_OPT_SATURDAY]	CS_OPT_ROWCOUNT=*rowcount*
CS_OPT_DATEFORMAT= [CS_OPT_FMTMDY | CS_OPT_FMTDMY | CS_OPT_FMTYMD | CS_OPT_FMTYDM | CS_OPT_FMTMYD | CS_OPT_FMTDYM]	CS_OPT_SHOWPLAN

Options	
CS_OPT_FIPSFLAG	CS_OPT_STATS_IO
CS_OPT_FORCEPLAN	CS_OPT_STATS_TIME
CS_OPT_FORMATONLY	CS_OPT_STR_RTRUNC
CS_OPT_GETDATA	CS_OPT_TEXTSIZE=*textsize*
CS_OPT_ISOLATION= [CS_OPT_LEVEL1 | CS_OPT_LEVEL3]	CS_OPT_TRUNCIGNORE
CS_OPT_NOCOUNT	

Using the DTM XA Interface with CICS

This section explains how to setup your CICS environment to use the DTM XA Interface. See also “XA configuration file for DTM XA Interface” on page 20 for information on creating an XA configuration file.

Building the switch-load file

Each RM defined in the CICS environment must provide an XA switch-load file. The switch-load file is a component of your CICS configuration; it is referenced in the XAD. It contains the RM’s name, a flag, a version number and a set of non-null pointers to the RM’s entry points, provided by the DTM XA Interface.

All of the Sybase XADs share a single switch-load file. You can build your Sybase switch-load file using the file *sybasexa.c*, which is located in:

$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics/switch

The following is a listing of *sybasexa.c*:

```
/*
 **
 ** sybasexa.c
 **
 ** The sybasexa routine references the Sybase xa
 ** switch structure named "sybase_TXS_xa_switch".
 ** The switch structure is part of the
 ** XA product library "libdtmxa.a".
 **
```

```
** See your CICS documentation for details on the
** switch-load file.
*/

#include <stdio.h>
#include <tmxa/xa.h>

extern struct xa_switch_t sybase_TXS_xa_switch;
extern struct xa_switch_t RegXA_xa_switch;
extern struct xa_switch_t *cics_xa_switch;

struct xa_switch_t *sybasexa(void)
{
    cics_xa_switch = &sybase_TXS_xa_switch;

    cics_xa_init();

    return(&RegXA_xa_switch);
}
```

This source code references the Sybase XA switch structure, which is global to the DTM XA Interface and defined as follows:

```
struct xa_switch_t sybase_TXS_xa_switch =
 {
 "SYBASE_SQL_SERVER",
 TMNOFLAGS,
 0,
 xa_open,
 xa_close,
 xa_start,
 xa_end,
 xa_rollback,
 xa_prepare,
 xa_commit,
 xa_recover,
 xa_forget,
 xa_complete
 };
```

The use of TMNOFLAGS specifies that the DTM XA Interface supports thread migration but does not support dynamic registration or asynchronous operations. For a description of these features, see the *X/Open CAE Specification (December 1991) Distributed Transaction Processing: The XA Specification*.

Compiling the switch-load file on IBM RISC System/6000 AIX

Compile *sybasexa.c* using the makefile *sybasexa.mk*, which is located in *$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics/switch*.

This is a listing of *sybasexa.mk*. Edit it to reflect your configuration.

```
SYB_LIBDIR = $(SYBASE)/$(SYBASE_OCS)/lib
SYBLIBS = -lxadtm -lct_r.so -lcs_r.so -ltcl_r.so -lcomn_r.so -lintl_r
          -lxdsxom

all : sybasexa.c xlc_r4  -bnoquiet -v -D_THREAD_SAFE \
 -I/usr/lpp/encina/include sybasexa.c  \
 -o sybasexa \
 -esybasexa \
 -L/usr/lpp/cics/lib \
 -L$(SYB_LIBDIR) \
 $(SYBLIBS) \
 -lcicsrt -ldce -lm \
 /usr/lpp/cics/lib/regxa_swxa.o
```

Note You must use the shareable versions of CS-Library (*libcs_r.so.*) and Common Library (*libcom_r.so.*).

Compiling the switch-load file on HP9000 Series 800 HP-UX

Compile *sybasexa.c* using the makefile *sybasexa.mk.hp800*, which is located in*$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics/switch*

This is a listing of *sybasexa.mk.hp800*. Edit it to reflect your configuration.

```
#
# Makefile to compile the LoadSwitchTable
# This makefile should be run with the command
# "make -f sybasexa.mk.hp800"
#

CC=/opt/ansic/bin/cc
CCOPTS= -Aa +z -Dsybasexa=CICS_XA_Init
ENCINA=/opt/encina
CICS=/opt/cics
LD=/usr/ccs/bin/ld

SYB_LIBDIR = $(SYBASE)/$(SYBASE_OCS)/lib
CICS_LIBDIR = $(CICS)/lib
```

```
all: sybasexa

sybasexa: sybasexa.o
$(LD) -b \
   +e CICS_XA_Init \
   -o sybasexa \
   sybasexa.o \
   $(CICS_LIBDIR)/regxa_swxa.o \
   -Bimmediate -Bnonfatal +s +b/opt/cics/lib \
   $(SYB_LIBDIR)/libxadtm.a \
   $(SYB_LIBDIR)/libct_r.a \
   $(SYB_LIBDIR)/libcs_r.sl \
   $(SYB_LIBDIR)/libtcl_v.a \
   $(SYB_LIBDIR)/libcomn_v.sl \
   $(SYB_LIBDIR)/libintl_r.sl \
   -lm \
   $(CICS_LIBDIR)/libcicsrt.sl \
   -lc

sybasexa.o: sybasexa.c
   $(CC) -c $(CCOPTS)\
   -I$(ENCINA)/include sybasexa.c
```

Note You must use the shareable versions of CS-Library (*libcs_r.sl*) and Common Library (*libcomn_dce.sl*).

You must have the ANSI C compiler to build the Load Switch Table.

Compiling the switch-load file on Sun Solaris 2.x (SPARC)

Compile *sybasexa.c* using the makefile *sybasexa_sol.mk* which is located in *$SYBASE/$SYBASE_OCS/sample_dtm/cics/switch*.

This is a listing of *sybasexa_sol.mk*. Edit it to reflect your configuration.

```
#Makefile to compile the LoadSwitchTable
#This makefile should be run with the command "make -f
sybasexa_sol.mk"

SYB_LIBDIR = $(SYBASE)/lib
SYBLIBS = lxadtm -lct_r -lcs_r.so -ltcl_r -lcomn_r.so
-lintl_r -lxdsxom -lm

all: sybasexa.c
/bin/xlc_r -v -D_THREAD_SAFE \
```

```
-I /usr/lpp/encina/include sybasexa.c \
-o sybasexa \
-esybasexa \
-L/usr/lpp/cics/lib \
-L$(SYBLIBS) \
-lcicsrt \
/usr/lpp/cics/lib/regxa_swxa.o -ldce
```

Adding a Sybase stanza to the CICS region XAD definition

The CICS TM uses CICS XAD information to communicate with other RMs. The XAD definition contains one Sybase stanza for each LRM. For a description of an XAD stanza's attributes, see your CICS documentation.

Below are two sample Sybase XAD stanzas. Use the SMIT utility to add stanzas to your CICS region.

```
betaOne:
GroupName=""
ActivateOnStartup=yes
ResourceDescription="XA Product Definition"
AmendCounter=2
Permanent=no
SwitchLoadFile="/usr/lpp/sybase/sample/xa_library/
                cics/switch/sybasexa"
XAOpen="-Uuser_1 -Ppassword_1 -Nconnection_1"
XAClose="ignored"
XASerialize=all_operations


betaTwo:
GroupName=""
ActivateOnStartup=yes
ResourceDescription="XA Product Definition"
AmendCounter=2
Permanent=no
SwitchLoadFile="/usr/lpp/sybase/sample/xa_library/
                cics/switch/sybasexa"
XAOpen="-Uuser_2 -Ppassword_2 -Nconnection_2"
XAClose="ignored"
XASerialize=all_operations
```

The following fields are configuration-dependent and must be modified:

- SwitchLoadFile

- XAOpen
- XAClose
- XASerialize

Note All Sybase stanzas can use the same switch-load file.

See "Open string parameters for DTM XA Interface" on page 18 for information about the contents specified in the XAOpen string of the XAD Definition.

Using the DTM XA Interface with Encina

This section describes how to assign an open string and initialize an RM for use with the Encina. See also "XA configuration file for DTM XA Interface" on page 20 for information on creating an XA configuration file.

Assigning an open string with *monadmin create rm*

The monadmin create rm command assigns each LRM a user name and password in the form of an *open string*. The user name and password allow the DTM XA Interface to control a particular connection's access to Adaptive Server resources. See "Open string parameters for DTM XA Interface" on page 18 for more information about the contents of the open string.

The following shows sample screen contents of a monadmin create rm session:

```
echo "Creating connection_1 resource manager record"
monadmin delete rm connection_1 >>& demo_conf.log
monadmin create rm connection_1\
-open "-Usa -Psecret -Nconnection_1" \
-close "not used" >>& \
demo_conf.log
if ($status) then
echo "Failed to create lrm_1 resource mgr.";
exit 1;
endif
```

Your current version of Encina may have additional commands for specifying RMs. For detailed information, see the *Encina Monitor System Administrator's Guide and Reference.*

Note You can use Encina enconsole interactive command instead of the shell monadmin command.

Initializing LRMs with *mon_RegisterRmi*

From within your Encina Monitor application server, you must register each LRM with a call to mon_RegisterRmi. For example:

```
status =
mon_RegisterRmi(&sybase_TXS_xa_switch,"connection_1",
&rmiID);
if (status != MON_SUCCESS)
{
fprintf(stderr, "mon_RegisterRmi failed (%s).\n",
mon_StatusToString(status));
bde_Exit(1);
}
fprintf(stderr, "mon_RegisterRmi complete\n");
```

For each LRM registered with a monadmin create rm command, there must be a mon_RegisterRmi command that initializes the LRM. The *rmname* specified in the monadmin create rm command must match the *rmname* in the mon_RegisterRmi command.

See the *Encina Monitor Programmer's Guide* for:

- Information about the tasks performed by the registration function and the order in which they must be performed
- Full syntax of the mon_RegisterRmi command

Linking applications with DTM XA Interface libraries

Link applications with the DTM XA Interface library, *libxadtm.a*.

Establishing connections

The TM, together with the DTM XA Interface library, establishes connections between applications and RMs in several steps:

1 An application issues a mon_RegisterRmi function, thereby requesting use of an LRM.

2 Using information contained in an open string, the TM issues an XA open call to the LRM (configured in the monadmin create rm command) whose name matches that issued in step 1, above.

3 The TM passes the open string associated with each monadmin create rm command to the DTM XA Interface. The open string contains the LRM name.

4 The DTM XA Interface looks up the LRM name in the XA configuration file and matches it to an actual RM name—that is, to an actual physical Adaptive Server. The RM name matches an entry in the Adaptive Server *interfaces* file.

5 The DTM XA Interface establishes one logical connection to an Adaptive Server for each LRM entry. It then confers on any connection the preconnection properties and capabilities configured for the LRM.

Using the DTM XA Interface with TUXEDO

The following sections explain the application-specific steps you need to take to integrate the XA Interface with TUXEDO.

The application-specific part of the integration involves:

- Linking the application with the application servers
- Setting up the *UBBCONFIG* file
- Building a transaction monitor server (TMS)
- Integrating the application servers with the resource managers

It is assumed that TUXEDO is installed in the *$TUXDIR* directory and that any resource managers are also installed on the system.

Note In the following procedures, replace the environment variables with the actual TUXEDO paths as follows: replace *$TUXDIR* with your actual root directory path, and replace *$SYBASE* with the path to the DTM XA Interface installation directory.

Table 3-7 provides the Sybase-specific information you need to perform the TUXEDO integration. The *TUXEDO Installation Guide* discusses this information in "Integrating a Resource Manager With System/T".

Table 3-7: Information needed to integrate the TUXEDO System

Type of Information	Sybase Specific	Description
RM name	SYBASE_XA_SERVER	The name of the resource manager in the *name* element of the xa_switch_t structure.
XA structure name	sybase_TUX_xa_switch	The name of the xa_switch_t structure that contains the resource manager identifier, the flags for the resource manager's capabilities, and the function pointers of the XA functions.
Library name	The library files *ct_r, cs_r, comn_r, tcl_r,* and *intl_r* which are located in *$SYBASE/$SYBASE_OCS/lib*	The list of files needed to support the DTM XA Interface, and a full path name.
Open string contents	See " Open string parameters for DTM XA Interface" in this document for more information..	The format of the information string passed to the functions.

Note The DTM XA Interface has been fully tested with the reentrant libraries, *ct_r*, *cs_r*, *comn_r*, *tcl_r*, and *intl_r*. If you are using the non-reentrant libraries and experience problems, use the reentrant versions of the libraries instead.

See also "XA configuration file for DTM XA Interface" on page 20 for information on creating an XA configuration file.

Linking

The TUXEDO RM file provides information used by TUXEDO utilities to link TUXEDO servers. Make sure that the RM file contains an appropriate set of specifications for linking Sybase applications.

1 Use the text editor of your choice to open the *$TUXDIR/udataobj/RM* file for editing.

2 Update the file with XA information by adding/verifying entries for Sybase resource managers. For most Sybase applications, including the simprpc.ct sample application, one entry for SYBASE_XA_SERVER is all that you need. If you are going to build and run the rentapp sample, you may want to go ahead and add the second entry for SCRAP_XA_SERVER, as required for rentapp.

Replace *$SYBASE/$SYBASE_OCS* with the fully qualified path to the Sybase installation directory containing the XA Interface:

```
SYBASE_XA_SERVER:sybase_TUX_xa_switch:-t -Bstatic -L$SYBASE/$SYBASE_OCS/lib
-lcobct -lxadtm -lct_r -lcs_r -lcomn_r -ltcl_r -lintl_r -Bdynamic -ldl
SCRAP_XA_SERVER:sybase_TUX_xa_switch:-t -Bstatic -L$SYBASE/$SYBASE_OCS/lib
-lcobct -lxadtm -lct_r -lcs_r -lcomn_r -ltcl_r -lintl_r -Bdynamic -ldl
```

Note Each entry *must* be a single continuous line.

The *cobct* libraries are only needed if you are building ESQL/COBOL application servers. If you are not using ESQL/COBOL, you can remove the -lcobct specification.

If you want your TUXEDO servers to load and execute all Sybase libraries dynamically, you can use entries like the following. Dynamic libraries may increase CPU overhead for TUXEDO server execution.

```
 SYBASE_XA_SERVER:sybase_TUX_xa_switch:-L$SYBASE/$SYBASE_OCS/lib -lxadtm
-lct_r -lcobct -lcs_r -lcomn_r -ltcl_r -lintl_r
 SCRAP_XA_SERVER:sybase_TUX_xa_switch:-L$SYBASE/$SYBASE_OCS/lib -lxadtm
-lct_r -lcobct -lcs_r -lcomn_r -ltcl_r -lintl_r
```

Note Each entry *must* be a single continuous line.

You can add a comment line by identifying it with a leading pound sign (#) character.

Setting up the UBBCONFIG file

This section provides specific examples for setting up the TUXEDO *UBBCONFIG* file with the XA Interface.

For the simprpc.ct sample application, the pubs2 database must be installed on Adaptive Server. Use the installation script in the Adaptive Server directory under *scripts/installpubs2*.

"Open string parameters for DTM XA Interface" on page 18 explains the open string in the *UBBCONFIG* file.

1 Use the ASCII text editor of your choice to open *$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct/ubbsimpct* for editing. The file is shown here with line numbers to facilitate the discussion:

```
1  *RESOURCES
2  IPCKEY          123456
3
4  MASTER          sybsite
5  MAXACCESSERS    5
6  MAXSERVERS      5
7  MAXSERVICES     10
8  MODEL           SHM
9
10 MAXGTT          5
11
12 *MACHINES
13 yourmachine     LMID=sybsite
14 TUXDIR="$TUXDIR"
15 APPDIR="$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct"
16 TLOGDEVICE="$SYBASE/$SYBASE_OCS/sample/xadtm/tuxedo/
   simprpc.ct/tuxlog"
17 TLOGNAME=TLOG
18 TUXCONFIG="$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo
   /simprpc.ct/tuxconfig"
19 ULOGPFX="$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct/ULOG"
20
21 *GROUPS
22 DEFAULT:        TMSNAME=simprpccttms TMSCOUNT=2
23
24 GROUP1          LMID=sybsite    GRPNO=1
25 OPENINFO="SYBASE_XA_SERVER: -Uuserid1 -Ppassword1 -Nconnection1"
26
27 *SERVERS
28 simpsrv         SRVGRP=GROUP1   SRVID=1
```

```
29
30 *SERVICES
```

2 Replace entries in the file with entries appropriate for your environment as shown in this table:

Line number	Entry	Replace with
13	*yourmachine*	Replace with the name of the machine that contains the XA Interface installation. Remember that the machine name is case-sensitive.
14	*$TUXDIR*	Replace with the actual TUXEDO root directory path.
15, 16, 18, 19	*$SYBASE/$SYBASE_OCS*	Replace with the XA Interface installation directory.
22	simprpccttms	This parameter is specific to the simprpc.ct example. In general, this parameter should relate to the value specified in the -o parameter of the buildtms command described on “Building the TMS” on page 38.
25	Open string parameters	See “Open string parameters for DTM XA Interface” on page 18 for more information.

Note See the *TUXEDO Installation Guide* for a detailed discussion of the *UBBCONFIG* file.

Creating the TUXEDO configuration file

Set the *$TUXCONFIG* environment variable to a value that matches the entry in *ubbsimpct* by issuing this command:

```
setenv TUXCONFIG $SYBASE/$SYBASE_OCS/sample/xa-
dtm/tuxedo/simprpc.ct/tuxconfig
```

Create a TUXEDO configuration file from the *UBBCONFIG* file by executing this command:

```
$TUXDIR/bin/tmloadcf -y ubbconfig_file_name
```

For this verification, using the simprpc.ct sample, replace *ubbconfig_file_name* with *ubbsimpct*.

Building the TMS

Build the transaction monitor server (TMS) by executing this command:

```
$TUXDIR/bin/buildtms -r SYBASE_XA_SERVER -o
$TUXDIR/bin/output_filetms
```

where *output_file* is a name you choose for the transaction monitor server program. It is helpful to append tms to the name as shown here, so it is easily identified. Choose a unique name for the program so that it does not conflict with TMS programs for other resource managers (TMS, TMS_D, and TMS_SQL are reserved).

For the simprpc.ct example verification, the *UBBCONFIG* file uses simprpccttms, which is line 18 in the table on page 10.

The program is stored in *$TUXDIR/bin* so that the TUXEDO System/T start program can find it.

Build COBOL runtime environment

In CICS transactions, COBOL transactions use the COBOL runtime, which you must modify to communicate with the Sybase XA environment.

To configure CICS to support Sybase XA COBOL transactions:

1 Log in as root.

2 Set the *COBDIR* environment variable to the directory path for the MicroFocus COBOL installation.

3 Set the *PATH* environment variable to include the MicroFocus COBOL binary directory.

4 Change directory to the *$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics* directories.

5 Run *xa_make_cobol_runtime*.

Warning! This script assumes that the CICS COBOL runtime file is installed in */usr/lpp/cics/v1.1/bin*. If you have installed CICS somewhere else, you must edit this script to reflect your installation.

This script builds a MicroFocus COBOL runtime environment with CICS and Sybase XA support. It allows CICS transactions written in COBOL to reference XA Interface and Open Client functions at run time. The script takes several minutes to run. For more information, see your CICS documentation.

Note You must use MicroFocus COBOL 3.1 or higher.

CHAPTER 4

Application Programming Guidelines

Embedded SQL and Client-Library applications must conform to certain coding constraints in order to function within the Sybase XA environment. This chapter summarizes these constraints and provides a Client-Library code fragment and two Embedded SQL code fragments.

Topic	Page
X/Open DTP versus traditional Sybase transaction processing	41
Transaction and connection management	42
Deallocate cursor function with Client-Library	45
Dynamic SQL	46
Getting a Client-Library connection handle	46
Multiple-thread environment issues	49
Linking with CT Library	53
Sample embedded SQL COBOL fragment	53
Sample embedded SQL C fragment	55

X/Open DTP versus traditional Sybase transaction processing

The X/Open DTP model of transaction processing differs substantially from the traditional Sybase model. The traditional Sybase TP environment is connection oriented. Programs set up connections directly between the application program and Adaptive Server using connection management SQL statements. In the XA Interface environment, the XA Interface, using LRMs, sets up connections for the application.

Table 4-1 summarizes the differences.

Table 4-1: Traditional TP and X/Open DTP model differences

Traditional TP model	X/Open DTP model
There is one or more transaction per client/server connection.	There is no notion of connections. Components communicate through interfaces.
Transactions are usually local, with each transaction confined to a single Adaptive Server.	Transactions are global. They span resource managers. The work done within a transaction is accomplished using more than one resource manager.
Each Adaptive Server is responsible for the recovery of the data it contains.	The transaction manager is responsible for recovering the data stored in all of the resource managers.

Transaction and connection management

Applications must pay special attention to commands related to:

- Transaction management
- Connection management
- The current connection

Note The XA Interface uses an ANSI default isolation level of 3. To minimize read-only locking, programs can set the transaction isolation level in the XA configuration file, or they can use select xxx from table noholdlock in individual SQL operations. See the *Transact-SQL User's Guide* for additional information on transaction isolation levels.

Transaction management

The CICS, Encina, or TUXEDO TM is responsible for transaction management. This includes creating a global transaction in which all of an application's work is either committed or rolled back. Consequently, applications cannot issue SQL statements that manage transactions.

Specifically, applications cannot invoke the following Embedded SQL commands:

- begin transaction

- commit
- rollback

Client-Library applications cannot execute (via ct_command, ct_dynamic, or ct_cursor) any of these Transact-SQL commands:

- begin transaction
- commit transaction
- rollback transaction
- set (chained, noexec, isolation, parseonly, statistics io, statistics time)
- save transaction

Note The application must recognize Adaptive-Server-detected errors, and abort or roll back the transaction through the TM. This is especially important for Adaptive Server detected-deadlocks.

Connection management

Applications rely on the Sybase XA environment for management of client/server connections. Connection management occurs transparently to the application. Consequently, Embedded SQL applications cannot invoke the following commands for XA-managed connections:

- connect
- disconnect

Client-Library applications cannot call these Client-Library commands using XA-managed connections:

- ct_close
- ct_con_alloc
- ct_con_drop
- ct_con_props
- ct_config with the parameters:
  - *CS_ENDPOINT*
  - *CS_EXPOSE_FMTS*

    - *CS_HIDDENKEYS*
    - *CS_MAX_CONNECT*
    - *CS_NETIO*
    - *CS_TRANSACTION_NAME*
- ct_connect
- ct_exit
- ct_getloginfo
- ct_init
- ct_options with the parameters:
    - *CS_OPT_CHAINXACTS*
    - *CS_OPT_FORCEPLAN*
    - *CS_OPT_FORMATONLY*
    - *CS_OPT_NOEXEC*
    - *CS_OPT_PARSEONLY*
    - *CS_OPT_STATS_IO*
- ct_remote_pwd
- ct_setloginfo
- *CS_OPT_STATS_TIME*

In addition, Client-Library applications cannot call these CS-Library commands:

- cs_ctx_drop (with global context handle)
- cs_objects (*CS_CLEAR*, *CS_SET*)

The current connection

The notion of a default connection, as described in the Open Client Embedded SQL documentation, does not exist in the Sybase XA environment. Consequently, applications must always explicitly specify a current connection.

There are two ways to specify the current connection in Embedded SQL. They are:

- The set connection command
- The *at connection name* clause

A current connection does not span transactions. For example, an application must reset the current connection after each CICS SYNCPOINTcommand or Encina onCommit command. To avoid confusion about the scope of the current connection, Sybase recommends that you use the *at connection_name* clause with all Embedded SQL statements.

Nontransactional connections

Applications can open and use nontransactional connections with the normal Open Client or Embedded SQL Interfaces. Operations on such connections do not participate in the transaction and are not committed or rolled back. They may be useful for queries of unchanging databases and updates of data which can be inaccurate.

Deallocate cursor function with Client-Library

Application programs use and reuse connections that have been allocated for them via the XA Interface. Sybase's implementation of cursors starting with SQL Server version 10.1 requires cursor structures on both the client (TM/RM program) side and the Adaptive Server side.

When a client explicitly deallocates a cursor, or when the client connection is closed, Adaptive Server deallocates the server cursor structures.

When the first iteration of a program opens or closes a cursor but the connection stays allocated (as it does with XA-Library), the second iteration of the same program fails, as it attempts to open the same cursor name. Adaptive Server informs us that it already has a cursor by this name at the same nesting level.

The application program must explicitly close and deallocate the cursor before it commits or aborts its transaction. This must be done in the transaction program that allocates the cursor. Embedded SQL records information about cursors which allows the XA Interface to perform the deallocation.

With Client-Library, you must handle error paths so that cursors are deallocated before a TM abort code is called. That is, if the open cursor works, deallocate it.

Use ct_cursor( ) with type CS_CURSOR_CLOSE and option CS_DEALLOC.

Dynamic SQL

The use of dynamic SQL statements has many characteristics in common with cursors, with the additional complexity that temporary stored procedures are sometimes placed into Adaptive Server. The use of dynamic SQL is not recommended in transactional applications, but if they are used, the following guidelines must be adhered to:

- In Embedded SQL use “Method 3: Prepare and Fetch with a Cursor” (see the ESQL document or a description of this method) if possible. When this method is used, Embedded SQL places information in the system which allows the XA Interface to locate and deallocate all dynamic SQL and cursors.
- In all other cases, the dynamic SQL statements and all associated cursors must be closed and deallocated to avoid adverse effects on other transactions. Any associated Client-Library command structures should be dropped to avoid memory leaks. See the Open Client and ESQL documentation for information on how to drop these command structures.

Getting a Client-Library connection handle

Obtaining a connection handle is an issue specific to Client-Library applications.

When the TM opens a connection to Adaptive Server, the XA Interface allocates a CS_CONNECTION structure for its own use. Once control passes to the application, that application must use the connection handle contained in this structure.

To get the connection handle, specify CS_GET for the cs_object routine's *action* parameter with an object type of CS_CONNECTION. cs_object's *objdata* parameter returns a structure containing a *connection* field. This field contains the handle to the CS_CONNECTION structure.

Warning! The XA Interface also allocates a CS_COMMAND structure whose handle is returned in the *command* field of the structure to which the *objdata* parameter points. An application cannot use this command handle, as the XA Interface continues to use this handle, itself.

The following code fragment demonstrates how to retrieve the handle to the CS_CONNECTION structure:

```
/*
 ** Arguments:
 **connection null-terminated name of the connection
 **(ESQL) or LRM connHloaded with the CS_CONNECTION
 ** handle if the lookup is successful
 **
 ** Returns:
 ** CS_SUCCEED connection handle found successfully
 ** CS_FAIL unable to find connection handle for given
 ** connection /#include <stdio.h> #include <strings.h>
 ** #include <cspublic.h>CS_RETCODE getConn(connection,
 ** connH)CS_CHAR connection[128];CS_CONNECTION connH;
 {
 CS_INT retcode;
 CS_CONTEXT *ctx;
 CS_OBJNAME name;
 CS_OBJDATA data;
 CS_THREAD thread_functions;
 CS_INT outlen;
 #define THREADID_SIZE 8
 CS_BYTE thread_id[THREADID_SIZE];
 /* Check arguments */
 if (strlen(connection) >= 128)
 {
 /* Connection name is too long */
 return(CS_FAIL);
 }
 /* Get the global context handle */
 retcode = cs_ctx_global(CS_VERSION_100, &ctx);
 if (retcode != CS_SUCCEED)
 {
 /* Major environment problems! */
```

```
return(CS_FAIL)
}
/*
** Initialize the CS_OBJNAME structure to look
** for the specified connection name.
*/
name.thinkexists = CS_FALSE;
name.object_type = CS_CONNECTNAME;
strcpy(name.last_name, connection);
name.fnlen = CS_UNUSED;
name.lnlen = CS_NULLTERM;
name.scopelen = CS_UNUSED;
/*
** Set the current thread-id so we get the instance of
** this connection that this thread should be using.
*/
retcode = cs_config(ctx, CS_GET,
CS_THREAD_RESOURCE, &thread_functions,
CS_UNUSED, &outlen);
if (retcode != CS_SUCCEED)
{
/*
** Even in an non-threaded environment,this should be
** successful.
*/
return(CS_FAIL);
}
name.thread = (CS_VOID *) thread_id;
retcode = (*thread_functions.thread_id_fn)(
name.thread, THREADID_SIZE,
&name.threadlen);
if (retcode != CS_SUCCEED)
{
return(CS_FAIL);
}
/*
** Initialize the CS_OBJDATA structure to receive the
** connection handle for this connection name
*/
data.actuallyexists = CS_FALSE;
data.connection = (CS_CONNECTION *) NULL;
data.command = (CS_COMMAND *) NULL;
data.buffer = (CS_VOID *) NULL;
data.buflen = CS_UNUSED;
/* Retrieve the connection information */
retcode = cs_objects(ctx, CS_GET, &name,
```

```
 &data);
 if (retcode == CS_SUCCEED)
 {
 if (data.actuallyexists == CS_TRUE)
 {
*connH = data.connection;
return(CS_SUCCEED);
}
else
{
/* No connection by that name exists */
return(CS_FAIL);
}
}
else
{
/*
** The global CS_CONTEXT handle is probably not
** initialized with connection information yet
*/
return(CS_FAIL);
}
}
```

Multiple-thread environment issues

Threads are multiple, simultaneous paths of execution in a single operating system process, and share access to the resources allocated to that process.

Some application programming interfaces (APIs) allow an application developer to effectively use threads in the transaction environment. In turn, Sybase's XA Interface supports a maximum level of concurrency, enabling it to take advantage of those environments.

However, this raises several issues for an application developer. For background information and a complete discussion of the issues, see the OSF's *DCE Application Developer's Guide*.

The *Open Client Reference Manual* contains a section on thread-safe programming. XA Interface assigns connections to threads at the request of the TM. These assignments ensure that only one thread at a time is working on the connection and is the reason the thread ID is included in the cs_object request described in "Getting a Client-Library connection handle" on page 46. As long as connections assigned by XA Interface are used in the thread to which they are assigned and the restrictions on their use are followed, there should be no Open Client or ESQL threading-related problems.

Caveats of thread use

Client-Library uses a connection state machine to verify that applications call Client-Library routines in a logical sequence. See Chapter 2, "Program Structure" in the *Open Client Client-Library/C Programmer's Guide* for an explanation of the steps involved in structuring a Client-Library application.

The assumption underlying the use of threads is that when a thread disassociates from a transaction branch, it leaves the state machine in an inactive state. By default, all Embedded SQL statements leave the connection quiescent. With Client-Library, this is true only in the following circumstances:

- When ct_results returns CS_END_RESULTS, or CS_SUCCEED with a result type of CS_CURSOR_RESULT.
- After an application calls ct_cancel with *type* as CS_CANCEL_ALL.
- When an application returns CS_CANCELED. The APIs that return CS_CANCELED include *ct_send()*, *ct_results()*, and *ct_get_data()*.

Warning! If connections are not left in an inactive state, the consequences may include transaction rollbacks, extra overhead as the XA Interface cleans up the connection (which may require full connection close and reopen), and the possible failure of subsequent transactions. In such a situation, XA Interface attempts to maintain application operation while it minimizes failure.

Embedded SQL thread-safe code

Thread-safe code protects the use of shared resources with a mutex (MUTual EXclusion semaphore). A mutex protects shared resources, such as files and global variables, by preventing them from being accessed by more than one thread at a time.

Use the -h (UNIX) or /threadsafe (VMS) precompiler option to generate thread-safe code.

Tightly coupled transactions

The XA environment treats each thread or process that works on a transaction as a transaction branch. Each transaction branch is assigned a different xid and works independently of the other branches. However, all branches are committed or rolled back as a unit.

Some TMs allow branches to be tightly coupled. Tightly coupled branches are assigned the same xid and work together on the transaction. In such cases the open string can contain the -O1 option. This option causes Adaptive Server to move the work among connections on demand and eliminates any lock that might otherwise occur between the connections. See your TM documentation to determine how the TM can be configured for tightly coupled operation.

Warning! Set the -O1 option only when the application design is guaranteed to avoid conflicting updates. Normally this is true only when the application branches are fully serialized, (branch B operates only after branch A completes). Data inconsistency may occur if the interaction of the tightly coupled branches is not well designed.

Without the -O1 option, attempts by the branches to update the same database row can result in a deadlock internal to the transaction. The -O1 option has no practical effect when the branches are not tightly coupled through the TM and are assigned different xids.

Warning! Cursors and dynamic SQL cannot be retained when the transaction is assigned to a different connection. Therefore, they should not be used unless the application structure guarantees that they are opened and closed during a period when no other branch will work on the transaction.

Note The transaction is reassigned to another connection only between SQL batches. A tightly coupled application can ensure that a set of operations is completed without conflict by performing all the operations in a single batch. This implies that operations within a single stored procedure are also completed without conflict.

For example, if row z in table B must contain the sum of rows x and y in table A. The following can result in an invalid value in row z:

```
Branch 1:                          Branch 2
    Updates Row x -> 5
    Reads Row y (= 4)
                                   Updates Row y -> 5
                                   Reads Row x (= 5)
                                   Updates Row z -> 10
    Updates Row z -> 9 (wrong value)
```

No problem occurs if the branches are performed serially:

```
Branch 1:                          Branch 2
    Updates Row x -> 5
    Reads Row y (= 4)
    Updates Row z -> 9
                                   Updates Row y -> 5
                                   Reads Row x (=5)
                                    Updates Row z -> 10
```

A control branch can also be used to resolve the problem:

```
 Branch 0:              Branch 1:           Branch 2
 (controller)
Starts Branches 1 and 2
Waits for both to complete
                     Updates Row x   Updates Row y -> 5
                      Terminates          Terminates
    Reads Row y
    Reads Row x
    Updates Row z
```

TM specific branch control mechanisms must be used to implement these serialization mechanisms.

Linking with CT Library

The XA Interface requires that the application be linked with the threaded versions of the Open Client Libraries. See the *Open Client/Server Supplement* for your platform to identify the libraries you must specify. If you do not link the proper thread-safe libraries, you may experience a variety of Open Client failures.

Sample embedded SQL COBOL fragment

This code fragment sets the current connection, and inserts data into an Adaptive Server database:

```
*REMARKS. TRANSACTION-ID IS 'POPS'.
*THIS TRANSACTION POPULATES A DATABASE'S DATA TABLE
*WITH STOCK DATA ENTRIES.


ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY DFHBMSCA.
COPY DFHAID.
COPY AIXCSET.
EXEC SQL INCLUDE SQLCA END-EXEC.
77  RESPONSE            PIC 9(8) COMP.
01  MSG-LIST.
02  MSG-1           PIC X(70) VALUE
'Transaction Failed: Unable To Prime Stock'
-'Table.'.
02  MSG-2           PIC X(70) VALUE
'Stock Records Added Successfully.'.
01  TRANSFAIL           PIC X(70).

EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  STOCK-RECORD.
02  STOCK-NUM       PIC X(5).
02  ITEM-DESC       PIC X(30).
02  STOCK-QTY       PIC X(7).
02  UNIT-PRICE PIC S9(4)V99 VALUE ZEROES.
EXEC SQL END DECLARE SECTION END-EXEC.

PROCEDURE DIVISION.
```

```
* CHECK BASIC REQUEST TYPE
*
IF EIBAID = DFHCLEAR
EXEC CICS SEND CONTROL FREEKB
END-EXEC
EXEC CICS RETURN
END-EXEC
END-IF.


* MAIN PROCESSING
*SET UP STOCK RECORD DETAILS AND THEN WRITE OUT
*STOCK RECORD.
*
MOVE '31421'TO STOCK-NUM.
MOVE 'Widget (No.7)'TO ITEM-DESC.
MOVE '0050035'TO STOCK-QTY.
MOVE 25.55 TO UNIT-PRICE.
PERFORM WRITE-STOCKREC.

MOVE '43567'TO STOCK-NUM.
MOVE 'Splunkett ZR-1'TO ITEM-DESC.
MOVE '0005782'TO STOCK-QTY.
MOVE 143.79 TO UNIT-PRICE.
PERFORM WRITE-STOCKREC.

EXEC CICS SYNCPOINT
RESP(RESPONSE)
END-EXEC.

IF RESPONSE NOT = DFHRESP(NORMAL)
MOVE MSG-1 TO TRANSFAIL
PERFORM FAIL-TRANS
END-IF.

MOVE MSG-2 TO MSGOUTO.
EXEC CICS SEND MAP('MSGLINE')
MAPSET('AIXCSET')
FREEKB
END-EXEC.
EXEC CICS RETURN
END-EXEC.
GOBACK.


* ATTEMPT TO WRITE OUT NEW STOCK RECORD.
```

```
*
WRITE-STOCKREC.
EXEC SQL SET CONNECTION connection_2
END-EXEC

IF SQLCODE NOT = 0
MOVE MSG-1 TO TRANSFAIL
PERFORM FAIL-TRANS
END-IF.

EXEC SQL INSERT INTO STOCK VALUES (:STOCK-RECORD)
END-EXEC

IF SQLCODE NOT = 0
MOVE MSG-1 TO TRANSFAIL
PERFORM FAIL-TRANS
END-IF.


* IF UNABLE TO APPLY CREATE, END TRANSACTION
* AND DISPLAY REASON FOR FAILURE.
*
FAIL-TRANS.
MOVE TRANSFAIL TO MSGOUTO
EXEC CICS SEND MAP('MSGLINE')
MAPSET('AIXCSET')
FREEKB
END-EXEC
EXEC CICS RETURN
END-EXEC.
```

Sample embedded SQL C fragment

This code fragment sets the current connection, and accesses data stored on Adaptive Server:

```
EXEC SQL INCLUDE sqlca;
int rcode;

EXEC SQL BEGIN DECLARE SECTION;
char name[15];
char supplier[30];
```

```
char supplier_address[30];
int order_quantity;

EXEC SQL END DECLARE SECTION;
main()
{
char errmsg[400];
char qmsg[400];
short mlen;

EXEC SQL WHENEVER SQLERROR GOTO :errexit;
EXEC SQL WHENEVER SQLWARNING GOTO :errexit
EXEC SQL WHENEVER NOT FOUND GOTO :errexit

/* Get addressability for EIB... */

/*
** Write record to CICS temporary storage queue...
*/

/* Send the first map */

EXEC CICS SEND MAP("PANEL1") MAPSET("UXA1")
FREEKB ERASE RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X001");

/* Receive the response */

EXEC CICS RECEIVE MAP("PANEL1") MAPSET("UXA1")
RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X002");

/* Select a record from the table based on user input. */

sprintf(name, "%s", panel1.panel1i.newnamei);
EXEC SQL SET CONNECTION connection_1;
EXEC SQL SELECT name, supplier, supplier_address,order_quantity
INTO
:name, :supplier, :supplier_address, :order_quantity
FROM cheese
WHERE name = :name;

/* Handle "no rows returned" from SELECT */
```

```
if (sqlca.sqlcode == 100)
{
sprintf(panel4.panel4o.messageo, "%s",
NOCHEESE);
EXEC CICS SEND MAP("PANEL4") MAPSET("UXA1") FREEKB ERASE RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X009");

EXEC CICS SEND CONTROL FREEKB;
EXEC CICS RETURN;

}
/* Fill in and send the second map */

memset ( &panel2.panel2o, '0',
sizeof(panel2.panel2o));
sprintf(panel2.panel2o.nameo, "%s", name);
sprintf(panel2.panel2o.supplo, "%s",supplier);
sprintf(panel2.panel2o.addresso, "%s",
supplier_address);
sprintf(panel2.panel2o.ordero, "%d", order_quantity);

EXEC CICS SEND MAP("PANEL2") MAPSET("UXA1")
FREEKB ERASE RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X003");

/* Receive the response */

if (panel2.panel2i.questi == 'y')

{

/* Send the third map... */
/* Receive the response... */
/* Update the database */

order_quantity = atoi(panel3.panel3i.newordi);

EXEC SQL UPDATE cheese
set order_quantity = :order_quantity
where name = :name;

/* Write a record to the temporary queue */

sprintf(qmsg, "%s", "The cheese table was updated");
```

```
mlen = strlen(qmsg);


EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1")
FROM(qmsg) LENGTH(mlen) RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X010");
}
else
{

/*
** The user does not wish to update so free the keyboard and return...
*/
}
/* Commit the update */

EXEC CICS SYNCPOINT RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X011");

/*
** Send the fourth map confirming successful update...
*/

EXEC CICS RETURN;
errexit:
fprintf(stderr,"error in cheeseland %d\n",sqlca.sqlcode);

/* Handle general errors */

sprintf(errmsg, "%.60s\n",sqlca.sqlerrm.sqlerrmc);
strncpy(panel4.panel4o.messageo, errmsg, 60);
sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);

/*
** Send the fourth map with appropriate message...
*/

/* Rollback the transaction */

EXEC CICS SYNCPOINT ROLLBACK;
EXEC CICS SEND CONTROL FREEKB;
EXEC CICS RETURN;
 }
```

Index

Symbols

$SYBASE/$SYBASE_OCS/sample 17
$SYBASE/sample/xa_library/CICS/switch directory 26, 28
/usr/lpp/cics/v1.1/bin
  and CICS COBOL runtime file 39

A

aborted transaction 7
access
  monadmin create rm 12
accessing
  Adaptive Server data 1
ACID test 3
  atomicity 3
  consistency 3
  durability 4
  isolation 4
Adaptive Server
  accessing data in 1
  DTM XA environment 11
  DTM XA interface 6
  RM 9
  XA interface 10
XA configuration file 19, 21
AP 5
  and symbolic names 11
  DTP 5
  purpose 5
  transaction boundaries 7
  TX interface and 6
API 49
application program
  See AP
Application servers
  linking 34
at connection name clause 45
atomicity 3, 7

B

backward compatibility 18
begin transaction 42
branch
  transaction 4
building
  TM server 38

C

calls 13, 33
  to LRMs 13
  TX 7
  XA 7
CICS
  TM v
CICS XAD
  LRMs 12
  stanzas 12
Client-Library 2
  accessing data with v
  coding constraints for vi, 2
  native interface 6
  pre-connection capabilities 12
client-server connection
  LRM 11
coding constraints
  Client-Library 2
  Embedded SQL 2
command handles 47
commands
  dbcc 2, 11
  enconsole 12, 32
  grant 19

initial xa_open () 18
mon_RegisterRmi 32
monadmin create rm 12, 31
monadmin create rm strings 13
sp_role 19
TM-specific 11
commit 43
two-phase 7
commit phase 7
commit transaction 43
commit transactions
TX Interface and 6
commitment protocol
TM and 6
committed transaction 3
communication
initiation 6
TM and 6
conceptual view
X/Open DTP model 5
configuration
files 9, 14, 33
of LRMs 23
XA configuration file 22
configuration file
Sybase XA 12
XA 11
XACONFIG FILE and 20
configuration files
contents of 12
LRM name 13
TM-specific 11
UBBCONFIG 13
XAD 14
connection handle 46
sample program 47
connections 20
and stored information 11
and X/Open DTP model 42
current 44
default 44
establishing 33
establishing and managing 3
in traditional SYBASE TP 41
stored information 12
X/Open DTP model 11
consistency 3
CS_COMMAND structure 47
CS_CONNECTION structure 46
cs_object 47
CS-Library commands
invalid 44
ct_command 43
ct_cursor 43
ct_dynamic 43
current connection 44

D

dbcc
XA-specific 2, 11
decisions
heuristic 8
default connection 44
Distributed Transaction Processing
See DTP
documents
related vi
DTM
licensing 1
DTM XA environment 9
DTM XA interface 6, 10
dtm_tm_role 19
DTP
definition of 4
environment v
key components 4
management v
RM 5
TM 4
X/Open Distributed Transaction Processing 1
XA model, graphic of 10
durability 4
dynamic registration 27

E

editing
UBBCONFIG file 36
XA configuration file 22

Embedded SQL 2
accessing data with v
and coding constraints vi, 2
invalid commands 42, 43
native interface 6
embedded SQL/C 11
embedded SQL/COBOL 11
enable dtm 17
enabling
TM transactions 1
Encina
monadmin create rm 12
enconsole 12, 32
environment
building runtime 38
DTM XA 9
multiple thread issues 49
XA 3
environment variables
XACONFIGFILE 20
establishing connections 33
CICS 13
Encina 13
TUXEDO 14

F

failure recovery 6
TM and 6
files
and threads 51
configuration 9, 12, 13, 14, 33
interfaces 14
libcom_r.so 28
libcomn_dce.sl 29
libcs_r.sl 29
libcs_r.so 28
libxadtm.a 32
switch-load 26
syb_xa_log 18
sybasexa.c 26
sybasexa.mk 28
UBBCONFIG 14, 36
XA configuration 14, 22
xa_config 14, 23
flags
trace 22
function
mon_RegisterRmi 13
xa_open() 20

G

global
identifiers 6
recovery 2
transaction 7, 14
global identifiers
TM and 6
global recovery 8
global transaction 4
commit 7
global transactions
AP and 5
logical connections 14
TM and 6
TX Interface and 6
X/Open DTP model 4
global variables
and threads 51
grant 19

H

handles
command 47
heuristic decision
conflicts 9
heuristic decisions 8
heuristic transactions
managing 11

I

identifier
global 6
initial xa_open () 18
initialization 32

initiation
communication 6
integrating TUXEDO 33
interface
DTM XA 10
native 5, 6, 10
TX 5, 6
XA 5, 6
interfaces file 14
invalid commands
CS-Library 44
Transact-SQL 43
isolation 4

L

libcom_r.so 28
libcomn_dce.sl 29
libcs_r.sl 29
libcs_r.so 28
library_names 34
libxadtm.a 32
licensing
DTM 1
linking application servers 34
local transaction 4
log file
syb_xa_log 18
log file entries 19
log file parameters 19
logfile_name 18
logical connections 14
logical resource manager
See LRM
LRM
and symbolic names 11
client-server connections 11
configuration of 23
connections 11
initializing 32
mon_RegisterRmi 13
lrm_name 18

M

makefile
sybasexa.mk 28
sybasexa.mk.hp800 28
management of transactions 42
migration
thread 27
mon_RegisterRmi 13, 32
monadmin create rm 12, 13, 31
monadmin create rm command 33
mutex and threads 51

N

native interface 6
Client-Library and 6
Client-Library calls 10
Embedded SQL and 6
illustration of 5

O

-O-1 19
-O1 19
onCommit command 45
Open Client libraries 10
open string 12, 18, 31, 33
strings 12
open strings 13

P

parameters
enable dtm 17
logfile_name 18
lrm_name 18
-O-1 19
-O1 19
open string 18
password 18
TMNOFLAGS 27
traceflags 18
username 18

-v11 18
password 12, 13, 14, 18, 31
permissions
monadmin create rm 12
prepare phase 7
transaction branch 7
protocol
transaction commitment 6
two-phase commit 7

R

README file 17
recovery 4, 8, 42
failure 6
global 2, 8
XA interface and 6
registration
dynamic 27
requirements vi
resource manager
See RM
RM 5
Adaptive Server and 5
X/Open DTP model 4
XA interface and 6
rm_name parameter 34
rm_structure_name parameter 34
rmname 32
roles
dtm_tm_role 19
rollback 43
rollback transaction 3, 7, 43
TX Interface and 6

S

sample programs
retrieving connection handles 47
section 19, 21
set connection command 45
shared resources
RM and 5
X/Open DTP model 4
SMIT utility 30
sp_role 19
standard
XA interface 1
stored information
connections 12
strings
open 12, 13, 31, 33
structures 46
switch structure 27
switch-load files 26
syb_xa_log 18
Sybase stanza
adding 30
SYBASE traditional TP
versus X/Open DTP model 42
Sybase XA configuration file 12
Sybase XA environment 9
sybasexa.c file 26
sybasexa.mk makefile 28
sybasexa.mk.hp800 makefile 28
symbolic names
and APs 11
and LRM 11
SYNCPOINT command 45

T

test
ACID 3
thread migration 27
threads 49
TM 4
accessing Adaptive Server data 1
CICS v
purpose 6
TX interface and 6
XA interface and 6
TM server
building 38
TMNOFLAGS 27
TM-specific commands 11
TM-specific configuration files 11
trace flag parameters 19
trace flags 22

traceflags 18
transaction 3
  atomicity 3
  branch 4, 7
  committed 3, 7
  consistency 3
  durability 4
  global 4, 7, 14, 42
  isolation 4
  limitation on 42
  local 4, 42
  management v, 42
  processing 4, 6, 7
  recovery 4
  rollback 3, 7
transaction boundaries
  AP and 7
  TX Interface and 6
transaction identifier 4
transaction operation 19
transaction processing model 9
Transaction Processing Monitor
  See TM
transactions
  XA interface and 6
Transact-SQL commands
  invalid 43
TUXEDO
  integrating 33
two-phase commit 7
  commit phase 7
  prepare phase 7
  transaction completion 7
TX
  calls 7
  interface 5, 6

U

UBBCONFIG file 13, 14
  editing 36
user name 12, 13, 14, 31
username 18
utilities
  SMIT 30

V

-V11 18

X

X/Open
  DTP 1
X/Open DTP
  functional model 5
X/Open DTP Model 4
X/Open DTP model
  component communication 6
  component interaction 6
  connections 11
  global transactions 4
  multiple application programs 4
  multiple resource managers 4
  shared resources 4
  vs SYBASE TP 42
XA
  calls 7
  DTP model 10
  interface 5, 6
  interface standards 1
XA configuration file 11, 14
  editing 22
XA environment
  components of 9
XA product definition
  See CICS XAD
xa_config 23
xa_config file 14
xa_open 20
XACONFIGFILE 20