# AIX CICS/6000 and Relational Database Management Systems Integration: Experiences with the XA Interface

Document Number GG24-4214-00

July 1994

International Technical Support Organization
San Jose Center

**Take Note!**

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xiii.

**First Edition (July 1994)**

This edition applies to IBM AIX CICS/6000 Version 1 Release 1 for use with the AIX Operating System Version 3 Release 2.5.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 471, Building 70B
5600 Cottle Road
San Jose, California 95193-0001

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

# Abstract

This document describes the integration of IBM AIX CICS/6000 Version 1 Release 1 and the following relational databases running on AIX Version 3 Release 2.5: DATABASE 2 AIX/6000, Informix Version 5, Oracle7, and Sybase System 10.

The document is written to help you understand and implement the XA interface as defined in the X/Open Distributed Transaction Processing Model. A working knowledge of IBM AIX CICS/6000 and relational database management systems is assumed.

The document provides detailed information on how to implement the XA interface in both a local and a remote configuration. It also provides guidelines on how to design an application in those configurations.

DS AX (154 pages)

# Contents

# Figures

# Tables

# Special Notices

| | |
|---|---|
| AIX | AIX/6000 |
| AIXwindows | CICS |
| CICS/6000 | DATABASE 2 AIX/6000 |
| RISC System/6000 | |

| | |
|---|---|
| MicroFocus COBOL | Micro Focus Company |
| DCE, Distributed Computing Environment | Open Software Foundation |
| Encina, Transarc | Transarc Corporation |
| Informix | Informix Corporation |
| Oracle, Oracle7 | Oracle Corporation |
| Sybase | Sybase, Inc. |
| X/Open | X/Open Company Limited |

# Preface

The X/Open Distributed Transaction Processing standard (X/Open DTP) defines the concept of resource managers coordinated by a transaction manager. Relational databases like DATABASE 2 AIX/6000, Informix, Oracle, and Sybase are typical resource managers. The transaction manager coordinates transaction initiation and completion among these resource managers.

CICS/6000 transactions can access relational databases by including embedded SQL calls inside an application program. Because CICS/6000 fulfills the role of a transaction manager in the X/Open DTP model, it can coordinate distributed transactions to XA-enabled relational databases. XA is the so-called interface between a transaction manager and resource managers in the X/Open DTP model.

This document describes the integration of IBM AIX CICS/6000 Version 1 Release 1 and the following relational databases running on AIX Version 3 Release 2.5: DATABASE 2 AIX/6000, Informix Version 5, Oracle7, and Sybase System 10. It provides detailed information on how to implement the XA interface in both a local and a remote configuration. It also provides guidelines on how to design an application in those configurations.

The document is written to help you understand and implement the XA interface as defined in the X/Open DTP model. A working knowledge of IBM AIX CICS/6000 and relational database management systems is assumed.

## How This Document Is Organized

The document is organized as follows:

- Chapter 1, “X/Open Distributed Transaction Processing Model”

  This chapter describes the functional and the process models as defined in the X/Open DTP standard.

- Chapter 2, “XA Integration Concepts”

  This chapter discusses the integration of CICS/6000 and XA-enabled relational database management systems (RDBMSs).

- Chapter 3, “Local Implementation”

  This chapter describes the implementation steps needed to integrate CICS/6000 with UNIX RDBMSs in the case where both CICS/6000 and the UNIX relational database reside on the same machine.

- Chapter 4, “Remote Implementation”

  This chapter describes the implementation steps needed to integrate CICS/6000 with UNIX RDBMSs in the case where both CICS/6000 and the UNIX relational database reside on separate machines.

- Chapter 5, “ Resource Manager Initialization at Region Startup: A Diagrammatic View”

  This chapter describes the steps involved in the startup of a CICS/6000 region when connecting to an XA-compliant RDBMS.

- Chapter 6, “Problem Determination”

  This chapter describes how to handle problem determination when CICS/6000 is using XA support for UNIX relational database systems.

- Chapter 7, “Application Considerations”

  This chapter describes some of our experiences configuring CICS/6000 in different application scenarios.

## Related Publications

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

- *AIX CICS/6000 Application Programming Guide*, SC33-0814
- *AIX CICS/6000 Planning and Installation Guide*, GC33-0816
- *AIX CICS/6000 Message and Codes*, SC33-0817
- *AIX CICS/6000 Problem Determination*, SC33-0818
- *AIX CICS/6000 Application Programming Reference*, SC33-0886
- *AIX CICS/6000 Customization and Operation Guide*, SC33-0931
- *IBM DATABASE 2 AIX/6000 Installation Guide*, GC09-1570
- *IBM DATABASE 2 AIX/6000 Administration Guide*, SC09-1571
- *IBM DATABASE 2 AIX/6000 Programming Guide*, SC09-1572
- *IBM DATABASE 2 AIX/6000 Command Reference*, SC09-1575
- *IBM DATABASE 2 AIX/6000 Messages and Problem Determination Guide*, SC09-1577
- *Informix UNIX Products Installation Guide Version 5.0*
- *Informix OnLine Administrator's Guide*
- *Informix TP/XA User Manual*
- *Informix Error Messages*
- *Oracle7 Server for IBM RISC/6000 Installation and Configuration Guide*
- *Oracle7 Server Administrator's Guide*
- *Oracle7 Server for Unix - Administrator's Guide*
- *Sybase SQL Server Installation Guide for AIX*
- *Sybase XA-Library Integration Guide for CICS/6000*
- *X/Open Distributed Transaction Processing Reference Model*
- *X/Open Distributed Transaction Processing: the XA Specification*
- *X/Open Distributed Transaction Processing: the TX (Transaction Demarcation) Specification*

## International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

> *Bibliography of International Technical Support Organization Technical Bulletins,* GG24-3070.

To get listings of ITSO technical bulletins (redbooks) online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

**How to Order ITSO Technical Bulletins (Redbooks)**

IBM employees in USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should their IBM branch office.

Customers may order hardcopy redbooks individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order redbooks in online format on CD-ROM collections, which contain the redbooks for multiple products.

## Acknowledgments

This document is the result of a residency project conducted at the International Technical Support Organization, San Jose Center, during 1994.

The project was designed and managed by:

**Guido De Simoni** International Technical Support Organization, San Jose Center.

The document was written by:

**Sebastiano Ardiri** IBM Italy

**Jasjit Vig** IBM United Kingdom.

Thanks to the following people for the invaluable advice provided in the production of this document:

**Peter Shum** IBM Toronto Laboratories

**Nancy Brown** Informix

**Alan Slutsky** Informix

**Deepak Puri** Oracle

**Kelly Pecoraro** Sybase.

In addition I would like to thank those who took time to review this document:

**Neil Kolban** IBM Dallas System Center

**Everton Lewis** IBM Hursley Laboratories

**Dave Spencer** IBM West Hursley.

Special thanks go to the following people:

**Lorna Conas** International Technical Support Organization, San Jose Center

**Rolf Lie** Informix

**Dan Mori** Oracle

**Josh Bersin** Sybase.

# Chapter 1. X/Open Distributed Transaction Processing Model

In this chapter we describe the X/Open Distributed Transaction Processing (DTP) model, its software components and interfaces, and the flow of control.[1]

## 1.1 Functional Model

Figure 1 is the basic X/Open DTP model of an application program environment for transaction processing.

The boxes are functional components; the connecting lines are interfaces among the components that X/Open has published or intends to publish. The arrows indicate directions in which control may flow. Data may flow in both directions across each interface.

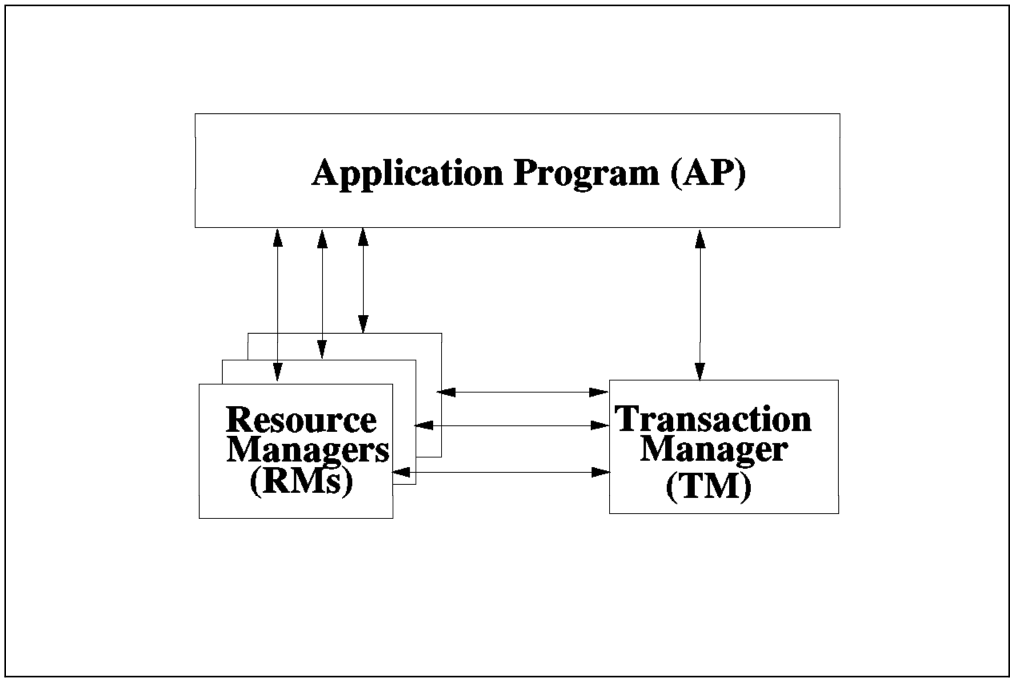


*Figure 1. X/Open Distributed Transaction Processing Model: Functional Components and Interfaces*

For a description of each functional component, see 1.1.1, “Functional Components” on page 2. For a description of the three interfaces shown, see 1.1.2, “Interfaces among Functional Components” on page 2.

The functional components are not necessarily separate processes, nor are they necessarily totally within a single process. For a description of the process model, see 1.2, “Process Model” on page 4.

[1] We extracted the information in section 1.1 of this chapter from pages 5-9 of *X/Open Distributed Transaction Processing Reference Model*, document number G120, ISBN 1 872630 16 2. We extracted the information in section 1.2 of this chapter from page 11 of the same book.

## 1.1.1 Functional Components

The three functional components of the X/Open DTP model are the application program, transaction manager, and resource manager.

#### 1.1.1.1 Application Program

The application program (AP) implements the desired function of the end-user enterprise. Each AP specifies a sequence of operations that involves resources such as databases. An AP defines the start and end of a global transaction, accesses resources within transaction boundaries, and usually decides whether to commit or roll back each transaction.

#### 1.1.1.2 Transaction Manager

The transaction manager (TM) manages global transactions and coordinates the decision to commit them or roll them back, thus ensuring atomic transaction completion. The TM also coordinates recovery activities of the resource managers when necessary, such as after a component fails.

#### 1.1.1.3 Resource Manager

The resource manager (RM) manages a certain part of the computer′s shared resources. Many other software entities can request access to the resource from time to time, using services that the RM provides. Examples of RMs include a database management system (DBMS), a file access method, such as X/Open indexed sequential access method (ISAM), or a print server. Some RMs manage a communication resource.

In the X/Open DTP model, RMs structure any changes to the resources they manage as recoverable and atomic transactions and let the TM coordinate completion of the transactions atomically with work done by other RMs.

## 1.1.2 Interfaces among Functional Components

There are three interfaces among the functional components in the basic X/Open DTP model:

- **AP-RM.** The AP-RM interfaces give the AP access to shared resources. Existing X/Open interfaces, such as structured query language (SQL) and ISAM, provide AP portability. The X/Open DTP model imposes few constraints on the native RM application program interface (API). X/Open may specify additional, specialized AP-RM communications interfaces for DTP, such as peer-to-peer and remote procedure call (RPC) interfaces.
- **AP-TM.** The AP-TM interface (the TX interface) lets the AP delimit global transactions. The TM provides routines that let the AP start and complete global transactions. The TM completes global transactions based upon a request from the AP and coordinates with the participating RMs and other involved TMs. When this coordination is completed, the TM returns the completion status of the AP. Details of the AP-TM interface are in *X/Open Distributed Transaction Processing: The TX (Transaction Demarcation) Specification*.
- **TM-RM.** The TM-RM interface (the XA interface) lets the TM structure the work of RMs into global transactions and coordinate global transaction completion and recovery. In the XA specification, the routines that each RM provides for the TM′s use are the *xa*_routines. The routines the TM provides for the RMs to call form the *ax*_set.

When an AP calls a TM through the TX interface, the TM typically implements each TX call by contacting RMs through the XA interface. Because the XA interface is invisible to the AP, the TM and RM may use other methods to interconnect without affecting application portability.

A TM assigns a data structure called a *transaction identifier* (XID). The XID lets the TM track and coordinate all of the work associated with a global transaction. Each RM maps the XID to the recoverable work it did for the transaction. For global uniqueness, the XID should contain atomic action identifiers.

### 1.1.3 Activity among Functional Components

The activity among the three functional components of the X/Open DTP model is related to different transaction operations.

#### 1.1.3.1 Transaction Initiation

When an AP instructs its TM to start a global transaction, the TM tells all appropriate RMs to associate the information about the global transaction with any work the AP may request from them.

Some RMs are configured so that the TM does not inform them when a global transaction starts. The RM contacts the TM to become associated with a global transaction only after the AP calls it to request actual work. This is called *dynamic registration*. If *static registration* is requested, each new transaction in the process generates a call to the RM that tells it that it has joined a new transaction or resumed an old one.

#### 1.1.3.2 Transaction Commitment

When an AP instructs its TM to commit a transaction, the TM and RMs use two-phase commit presumed rollback to ensure transaction atomicity.

In Phase 1, the TM asks all RMs to *prepare to commit* (or *prepare* ) their work. It also asks whether the RM can guarantee its ability to commit the work it did on behalf of a global transaction. If an RM can commit its work, it replies affirmatively. A negative reply reports failure.

In Phase 2, the TM directs all RMs either to commit or to roll back the work done on behalf of a global transaction, as the case may be. All RMs commit or roll back changes to shared resources and then return status to the TM.

When an AP calls its TM to commit a global transaction, the TM reports on whether commitment or rollback was the outcome. This report is based on reports the TM received (directly or through other TMs) from all involved RMs.

The XA specification contains two optimizations in the calling sequence between the TM and RM. An RM can withdraw from further participation in a global transaction during Phase 1 if it was not asked to update shared resources (the *read-only optimization*). A TM can use one-phase commit if it is dealing with only one RM that is making changes to shared resources.

The XA specification discusses requirements for stable recording of transaction data, including specifying when the TM and RMs are free to discard their knowledge of the global transaction.

### 1.1.3.3 Transaction Rollback

The TM rolls back the global transaction if any RM responds negatively to the Phase 1 request, or if the AP directs the TM to roll back the global transaction.

The TM effects Phase 2 by telling RMs to roll back the transaction. The RMs must not let any changes to shared resources become permanent.

### 1.1.3.4 Heuristic Transaction Completion

In certain, unusual, cases, the RM could experience a long delay between Phase 1 and 2 of the two-phase commit protocol. For example, the TM that issued the prepare-to-commit (Phase 1) request could block or fail later in the protocol sequence. In order to free resources, an RM may complete its work *heuristically* (independent of direction from its TM). The heuristic decision may be prompted by administrative action or by completion of a parametric timeout interval.

Heuristic decisions typically cannot depend on the knowledge of results at other RMs that TMs normally coordinate. When any RM makes a heuristic decision, the global transaction may fail to maintain global atomicity; one RM may commit its work while another rolls back its work. Such *mixed-heuristic* completion may leave shared resources in an inconsistent state. A TM may report a mixed-heuristic condition to its AP.

### 1.1.3.5 Failures and Recovery

Recovery is a process of restoring shared resources to a consistent state after various types of failure. The X/Open DTP model makes these assumptions:

- TMs and RMs have access to stable storage.
- TMs initiate and control transaction recovery.
- RMs provide for their own restart and recovery as directed by TMs.

## 1.2 Process Model

This section describes the actual processes of the DTP model (see Figure 2).

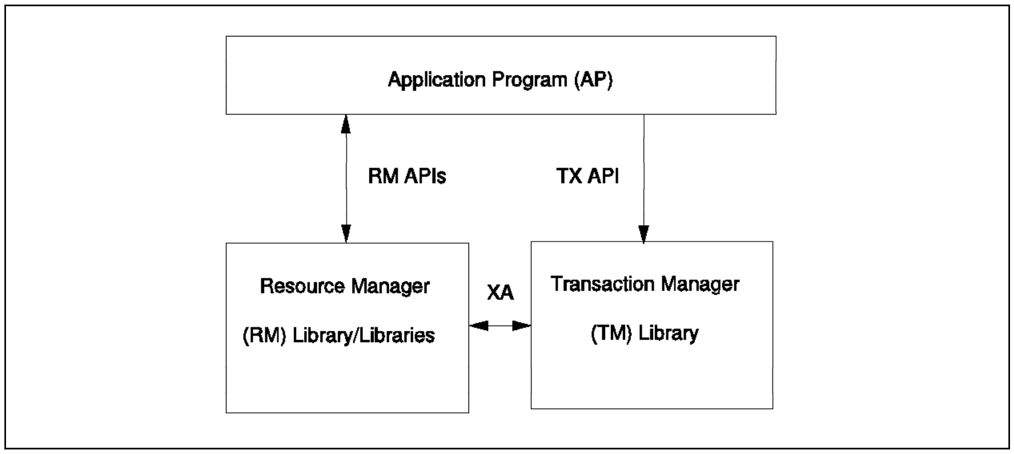

Application Program (AP)
RM APIs
TX API
Resource Manager (RM) Library/Libraries
XA
Transaction Manager (TM) Library

*Figure 2. X/Open Distributed Transaction Processing Model: Processes*

An instance of the model is an operating-system process, implemented by combining an AP with libraries for a TM and for one or more RMs. Through these libraries routines, the TM and RMs supply the AP with X/Open-compliant

APIs. The libraries also include interfaces between the TM and RMs with which the AP is not directly concerned.

The interface routines in an RM library include:

- The native interface routine that the AP calls
- The XA interface routines that the TM calls.

The interface routines in a TM library include:

- The TX interface routines that the AP calls
- The XA interface routines that some RMs may call.

Within a single instance of the model, an AP deals with exactly one TM, but the AP and TM may deal with multiple RMs.

## 1.3 CICS/6000 and the X/Open DTP Model

Figure 3 shows the architectural positioning of an IBM AIX CICS/6000 Version 1 Release 1 (CICS/6000) system in the X/Open DTP model.

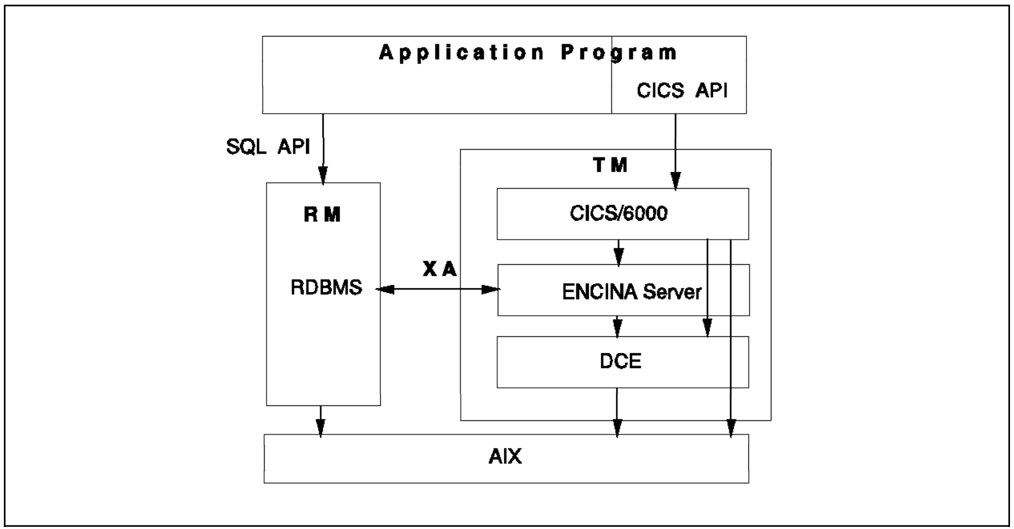


*Figure 3. CICS/6000 and the X/Open DTP Model: Architectural Positioning*

The CICS/6000 architecture hides the TX Interface.

# Chapter 2. XA Integration Concepts

In this chapter we discuss the integration of CICS/6000 and XA-enabled relational database management systems (RDBMSs). We divide the discussion into two parts:

- Relational database concepts
- CICS/6000 concepts.

## 2.1 Relational Database Concepts

In this section we introduce the concepts of:

- Shared object or library
- XA open string
- Resource manager switch
- Database privileges.

### 2.1.1 Shared Object or Library

CICS/6000 has an advanced AIX load-based architecture under which transaction code is dynamically loaded into long-running application server programs. This architecture allows new transactions to be added to a running CICS/6000 system. Because applications do not need to be relinked if there is an internal library change, application maintenance is easy.

To support the architecture, the vendors of XA-enabled RDBMSs supply shared reentrant versions of their objects or libraries for use with CICS/6000. These libraries are discussed individually in Chapter 3, "Local Implementation" on page 15 and Chapter 4, "Remote Implementation" on page 49.

### 2.1.2 XA Open String

A transaction manager calls *xa_open()* to initialize an RM and prepare it for use in a DTP environment.

Any information needed for the initialization of the RM (such as opening of files, identifying a resource domain) can be provided as parameters to *xa_open()*. For example, you need to provide DATABASE 2 AIX/6000 with a database name in order to connect to it.

The XA open string is one of the arguments to *xa_open()* an d is specific to the database. We discuss the XA open strings for the different databases in Chapter 3, "Local Implementation" on page 15 and Chapter 4, "Remote Implementation" on page 49.

If the resource manager supports multiple instances, the transaction manager can call *xa_open()* more than once for the same resource manager.

## 2.1.3 Resource Manager Switch

The XA interface provides two types of services: *ax_* services and *xa_* services.

Table 1 shows the *ax_* services, which:

- The TM provides
- Allow an RM to call a TM
- Allow an RM to dynamically control its participation in a transaction branch.

*Table 1. ax_ services in the XA Interface*

| Name | Description |
|---|---|
| *ax_reg* | Register an RM with a TM. |
| *ax_unreg* | Unregister an RM with a TM. |

Table 2 shows the *xa_* services, which:

- The RM provides
- Allow a TM to call an RM
- The TM calls in a particular sequence
- Allow a TM to inform an RM of transaction branch startup, commit, rollback, or coordinate failure recovery.

These are listed in Table 2

*Table 2. xa_ services in the XA Interface*

| Name | Description |
|---|---|
| *xa_close* | Terminate the AP's use of an RM. |
| *xa_commit* | Tell the RM to commit a transaction branch. |
| *xa_end* | Dissociate the thread from a transaction branch. |
| *xa_forget* | Permit the RM to discard its knowledge of a heuristically completed transaction branch. |
| *xa_open* | Initialize an RM for use by an AP. |
| *xa_prepare* | Ask the RM to prepare to commit a transaction branch. |
| *xa_recover* | Get a list of XIDs the RM has prepared or heuristically completed. |
| *xa_rollback* | Tell the RM to roll back a transaction branch. |
| *xa_start* | Start or resume a transaction branch; associate an XID with future work that the thread requests of the RM. |

RMs must provide information that gives the TM access to the *xa_* services. This information is provided through a *switch*, which is an external C variable of type **xa_switch_t**. The *switch* returns values to the TM. The return fields are:

| **Field** | **Description** |
|---|---|
| **name** | Resource manager name |
| **flags** | Options specific to the database. |

**version** value must be 0

**others** pointers to the *xa_* routines of the RM

Let′s have a look at the flag options.

**TMNOFLAGS** This flag specifies that no flags have been set, that is, the RM does not support dynamic registration or asynchronous mode. It supports association migration.

**TMREGISTER** This flag implies that the resource manager supports *dynamic registration*.

**Dynamic registration**: Certain RMs, especially those involved in relatively few global transactions, may ask the TM to assume that they are not involved in a transaction. For example, DATABASE 2 AIX/6000 supports dynamic registration. Therefore, like all other RMs, DATABASE 2 AIX/6000 is initialized by CICS/6000 at region startup by the *xa_open()* routine. However, CICS/6000 subsequently never calls DATABASE 2 AIX/6000 with the *xa_start()* routine.

**Note**

For all other RMs that do not support dynamic registration CICS/6000 involves all associated RMs in a transaction branch. CICS/6000 issues the *xa_start()* routine for all transactions even if one of those other RMs does not participate in the transaction. This operation could create an overhead if you have transactions accessing none or a few RMs.

When an AP requests works from a dynamically registered RM, before doing the work, the RM contacts the TM by calling the *ax_reg()* service. For more information refer to *X/Open Distributed Transaction Processing: The XA Specification*.

**TMNOMIGRATE** This flag indicates that the resource manager does not support *association migration*.

**Association migration**: Several threads may participate in a single transaction branch, some more than once. The *xa_start()* and *xa_end()* routines pass an XID to an RM to associate or dissociate the calling thread with a branch. Certain calls to *xa_end()* suspend the thread′s association. The call may indicate that the association can *migrate*, that is, that any thread may resume the association. In this case, the calling thread is no longer associated. For more information refer to *X/Open Distributed Transaction Processing: The XA Specification*.

**TMUSEASYNC** This flag indicates that the RM should use the *asynchronous mode* of operation for the *xa_* services.

**Aynchronous mode**: The *xa_* services typically operate synchronously, that is, control does not return to the caller until the operation is complete. Most *xa_* services have a form by which the caller requests asynchrony. Asynchronous calls should return immediately. The caller can subsequently call *xa_complete()* to test the aynchronous operation for completion. For more information refer to *X/Open Distributed Transaction Processing: The XA Specification.*

### 2.1.4 Database Privileges

RMs have different ways of implementing security. Somes RM require that certain database or table privileges be set before CICS/6000 can connect to them. We discuss these privileges for each XA-enabled RM in Chapter 3, “Local Implementation” on page 15 and Chapter 4, “Remote Implementation” on page 49.

## 2.2 CICS/6000 Concepts

In this section we introduce the concepts of:

- XA resource definitions
- CICS/6000 region environment setup
- CICS/6000 region startup.

### 2.2.1 XA Resource Definitions

CICS/6000 uses the product definitions to hold information it needs to interface to other transactional products, using the X/Open XA protocol. Each XA definition (XAD) entry contains information for one product. These entries can be found in a file called **XAD.stanza** in the directory ***/var/cics_regions/***<*region_name*>***/database/XAD***, where <*region_name*> is the name of the CICS/6000 region.

Each XAD entry consists of several attributes:

|  |  |
|---|---|
| <*Key*> | Represents the name of the product and is the key for the XAD entry. Product names can be up to 12 characters long. |
| **GroupName** | Group to which resource belongs. This 8-byte ASCII text attribute assigns a group name to which this resource description entry belongs. The default value is ″″. |
| **ActivateOnStartup** | If you set this attribute to yes, CICS/6000 always copies the resource definition from the permanent database to the run-time database at cold start. The default is *yes*. |
| **AmendCounter** | Number of updates (internal use only). Reserved for CICS/6000 internal use. |
| **Permanent** | Specifies whether or not CICS/6000 permits you to amend or delete the permanent database entry. If you set the attribute to *no*, you can amend or delete the entry. The default value is *no*. |
| **SwitchLoadFile** | Switch Load file path name. This ASCII text attribute is a path name to an object file (linked using ld but not a main() program) that contains the xa_switch_t structure definition and xa support subroutines for this XA-compliant product. The default value is ″″. |
| **XAOpen** | Resource manager initialization string, an ASCII character string. It is described as the *XA open string* in 2.1.2, “XA Open String” on page 7. It is specific to the RM. The default is ″″. |
| **XAClose** | Resource manager termination string. This is an ASCII character string that is passed to the XA-compliant product′s xa_close_entry() function call. The layout of the string is specific to the XA product being defined. The default value is ″″. |

**XASerialize** Resource manager serialization attribute. Indicates how CICS/6000 should serialize access to an XA-compliant RM in a multithread process. This attribute is used to indicate that an XA-compliant resource manager supports xa calls from multiple threads, and which style of serialization the XA-compliant product requires. The default value is *all_operations*. The following values are accepted:

**all_operations** CICS/6000 serializes around each xa call made. This is the default setting.

**start_end** CICS/6000 serializes around xa_start and xa_end calls. Serialization takes place around a transaction.

**single_association** or **multiple_association** Allows CICS/6000 to serialize based on whether the XA-compliant product is thread aware or not.

### 2.2.2 CICS/6000 Region Environment Setup

The file *environment* in the */var/cics_regions/<region_name>* directory, where *<region_name>* is the name of the CICS/6000 region, contains a list of environment variables that are set before the startup of the region. These variables may be specific to the RM to which you are connecting.

### 2.2.3 CICS/6000 Region Startup

Below we present an overview of the steps involved in the startup of a CICS/6000 Region when connecting to an XA-compliant RDBMS. We discuss the steps up to generic CICS/6000 initialization only. RM specific initialization is discussed in Chapter 5, " Resource Manager Initialization at Region Startup: A Diagrammatic View" on page 61.

**Note**

The RM must be running at CICS/6000 region startup. If the RM is not running, the region startup will fail on the *xa_open()* call.

Figure 4 on page 12 shows the flow of the CICS/6000 region startup.

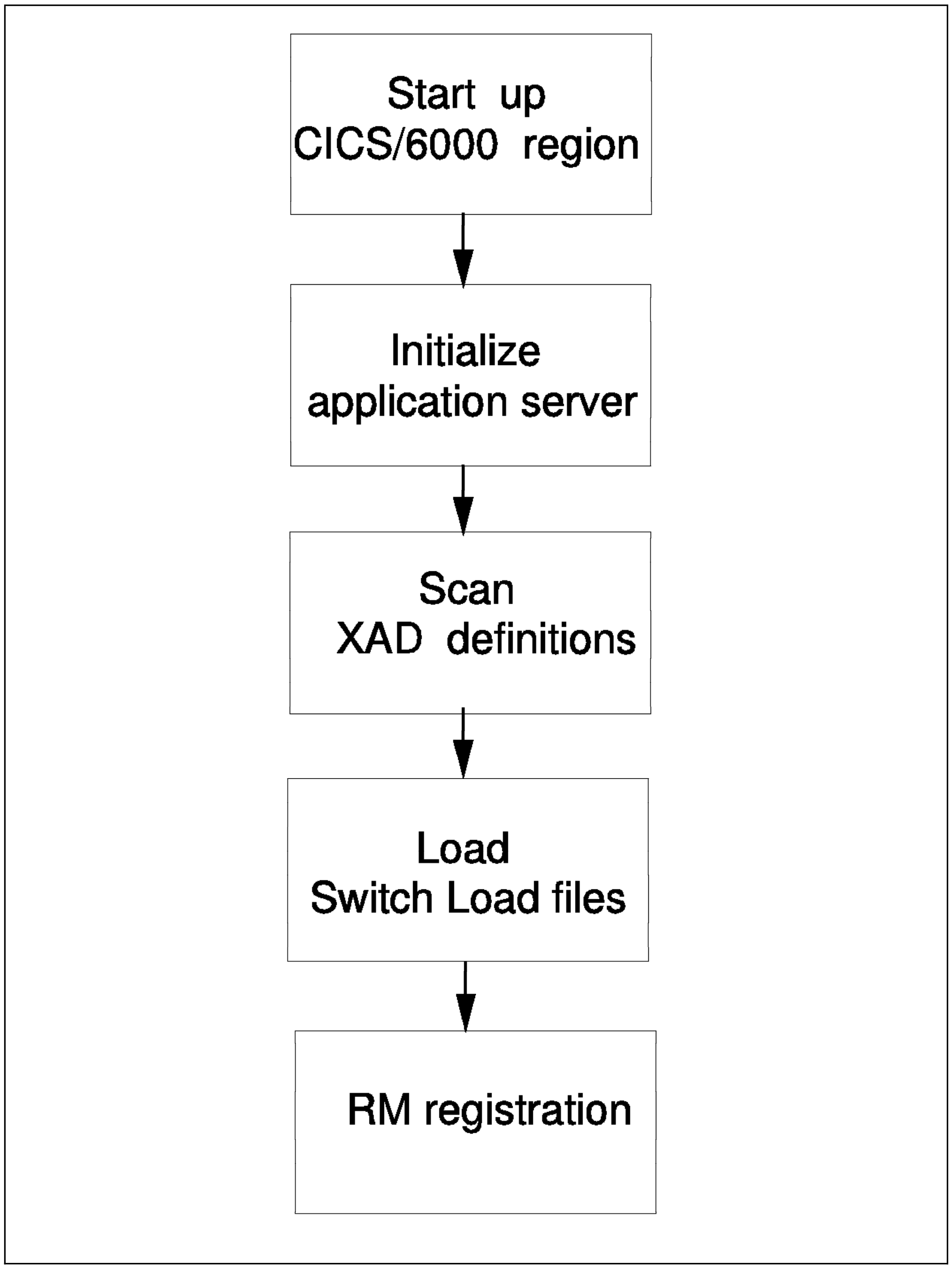


*Figure 4. CICS/6000 Region Startup*

The steps in the process are discussed as follows:

1. Start up CICS/6000 region

   Enter the following command:

   ```
   $ smitty cics
      └→ Manage CICS/6000 Regions
          └→ Cold Start a CICS/6000 Region
   ```

   or use the fastpath command: *smitty cicscoldstart*.

In the first phase of the startup the *cics* process initializes the *cicsld*, *cicsam*, *cicsrm*, and *cicsic* processes. In the XA implementation we are interested only in the *cicsam* initialization.

2. Initialize application server (*cicsam*)

   The application server registers its interfaces with *cicsts*.

3. Scan XAD definitions

   As part of the XA interface registration the XA definitions are scanned to obtain the Switch Load File name and the XA Open String of the RMs.

4. Load Switch Load files

   The Switch Load file as discussed in 2.2.1, "XA Resource Definitions" on page 10 is an object file. It is loaded into a CICS application server using the AIX *load()* system call. The object file must be linked such that a function call is the entry point. This function entry returns the address of an *xa_switch_t* structure (see 2.1.3, "Resource Manager Switch" on page 8).

5. RM registration

   CICS/6000 issues an *xa_open()* to the RM using the XA open string to prepare it for use in a distributed processing environment. Details of each RM registration are discussed in Chapter 5, " Resource Manager Initialization at Region Startup: A Diagrammatic View" on page 61.

# Chapter 3. Local Implementation

In this chapter we describe the implementation steps needed to integrate CICS/6000 with UNIX RDBMSs. We deal with the case where both CICS/6000 and the UNIX RDBMSs reside on the same machine.

We explain the implementation of four UNIX RDBMSs:

- DATABASE 2 AIX/6000
- Informix Version 5
- Oracle7
- Sybase System 10.

## 3.1 Before You Start

Check the following:

- AIX 3.2.5 is installed and running on your RISC System/6000.
- The following CICS/6000 Licensed Program Products (LPPs) are installed:
  - cics6000base.base.obj 1.1.0.0
  - cics6000.dev.obj 1.1.0.0
  - cics6000.prod.obj 1.1.0.0
  - cics6000clt.dev.obj 1.1.0.0
  - cics6000clt.prod.obj 1.1.0.0
  - cics6000mEn_US.msg 1.1.0.0.
- The following Encina LPPs are installed:
  - encServ.obj 1.1.1.0
  - encServmEn_US.msg 1.1.1.0
  - encExec.obj 1.1.1.0
  - encExecmEn_US.msg 1.1.1.0
  - encSfs.obj 1.1.1.0
  - encSfsmEn_US.msg 1.1.1.0.
- The following DCE LPPs are installed:
  - dcebase.base.obj 1.2.0.0
  - dcebase.En_US.msg 1.2.0.0
  - dcebase.admin.obj 1.2.0.0
  - dcebase.appdev.obj 1.2.0.0
  - dcecds.obj 1.2.0.0
  - dcecds.En_US.msg 1.2.0.0
  - dcesec.obj 1.2.0.0
  - dcesec.En_US.msg 1.2.0.0
  - dcepthreads.obj 1.1.0.0
  - dcepthreads.En_US.msg 1.1.0.0.

- The following PTFs have been applied:
  - CICS/6000 December PTFs: U420315, U420263, U422258, U421442
  - DCE base U419616
  - DCE threads U422532.
- DCE is configured on your machine as either a client or a server.
- Encina Structured File Server and logserver are configured and running.
- A CICS/6000 region has been created.

**Note**

We ran the XA interface with CICS/6000 Version 1, Release 1, Level 0. Informix Version 5.02 and Sybase System 10 are officially supported with CICS/6000 Version 1, Release 1, Level 1.

For more information on configuration requirements refer to the *AIX CICS/6000 Planning and Installation Guide*.

## 3.2 DATABASE 2 AIX/6000

We divide our discussion of the DATABASE 2 AIX/6000 implementation into the following parts:

- DATABASE 2 AIX/6000 configuration
- CICS/6000 configuration.

### 3.2.1 Before You Start

Make sure that the following DATABASE 2 AIX/6000 products are installed on your machine:

- IBM DATABASE 2 AIX/6000
- IBM AIX DATABASE 2 Client Application Enabler/6000
- IBM AIX DATABASE 2 Software Developer′s Kit.

Make sure that you have defined an instance for DATABASE 2 AIX/6000.

To create an instance:

1. Create an AIX userid. This is the instance owner ID.
2. Run the following command: */usr/lpp/db2_01_01_0000/instance/db2instance instance_name*, where *instance_name* is the AIX user created in step 1.
3. Edit the *.profile* file of the instance owner and add the following line: *.$HOME/sqllib/db2profile*.

For more information on how to create an instance refer to the *IBM DATABASE 2 AIX/6000 Administration Guide*.

## 3.2.2 DATABASE 2 AIX/6000 Configuration

The items that you need to consider in the implementation of DATABASE 2 AIX/6000 are:

1. Shared object
2. XA open string
3. Resource manager switch
4. Database privileges.

We deal with each of these in the sections that follow.

### 3.2.2.1 Shared Object

You need to create a DATABASE 2 AIX/6000 shared object code. As its name suggests, the shared code is loaded into memory once in the shared library segment and shared by all processes that reference it. The CICS/6000 COBOL run-time environment, CICS/6000 C transactions, and the Switch Load file need to reference the DATABASE 2 AIX/6000 shared object at run time.

The DATABASE 2 AIX/6000 shared object *db2.o* must be used as input when linking CICS/6000 C programs to ensure that both the CICS/6000 system and the CICS/6000 C programs share the same copy of DATABASE 2 AIX/6000.

The steps to create the DATABASE 2 AIX/6000 shared object are:

1. AIX login as **root**
2. *cd /usr/lpp/db2_01_01_0000/lib*
3. *ar -vx libdb2.a*
4. *mv shr.o db2.o*.

**Symbolic Links**

If, after installing DATABASE 2 AIX/6000, you used the *db2ln* script to create symbolic links from */usr* to the DATABASE 2 AIX/6000 library and include files, you need to add another symbolic link from */usr* to *db2.o* as follows:

*ln -s /usr/lpp/db2_01_01_0000/lib/db2.o /usr/lib/db2.o*.

These symbolic links allow your programs to be linked without any reference to a specific release of DATABASE 2 AIX/6000. If you do not use these symbolic links, you will have to relink your programs when you migrate to a newer release of DATABASE 2 AIX/6000.

### 3.2.2.2 XA Open String

We define the syntax of the XA open string in this section.

The XA open string must have the following syntax:

*database_alias*, < *username,password*> .

where the words in italics are values that you input.

The field values are:

**database_alias** Is the database name unless you have explicitly cataloged an alias name after database creation.

**username** Indicates a valid AIX userid. This is optional. It is used to provide authentication information to the database if the database is set up with *authentication=server*.

**password** This is the password for the above-mentioned userid.

**Note**

Because DATABASE 2 AIX/6000 uses a database name in the XA open string, you will need to:

- Create the database before region startup.
- Define an XA open string for each database that you use in a CICS/6000 application.

#### 3.2.2.3 Resource Manager Switch

Figure 5 shows the source code for the Switch Load file for DATABASE 2 AIX/6000. You can find this file in the */usr/lpp/cics/v1.1/src/examples/xa* directory, *db2xa.c*.

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

extern struct xa_switch_t db2xa_switch;
extern struct xa_switch_t RegXA_xa_switch;
extern struct xa_switch_t *cics_xa_switch;

struct xa_switch_t *CICS_XA_Init(void)
{
   cics_xa_switch = &db2xa_switch;

   cics_xa_init();

   return(&RegXA_xa_switch);
}
```

*Figure 5. Source Code for DATABASE 2 AIX/6000 Switch Load File*

Build the Switch Load file object, **db2xa**, by issuing the following command:

*make -f db2xa.mk*.

Figure 6 on page 19 shows the source code for the db2xa.mk makefile. You can also find the code for this makefile in the */usr/lpp/cics/v1.1/src/examples/xa* directory, *db2xa.mk*.

This makefile assumes you have run *db2ln* (see the *IBM DATABASE 2 AIX/6000 Installation Guide* for more details).

```
all: db2xa.c
    xlc_r -v -I/usr/lpp/encina/include \
    db2xa.c \
    -o db2xa \
    -eCICS_XA_Init \
    -L/usr/lpp/cics/v1.1/lib \
    -L/usr/lpp/encina/lib \
    -L/usr/lib \
    -lregxart -lsupprrt -linfdu -linftrrt -lsuperrt -ltasta -ltaslu \
    /usr/lpp/cics/v1.1/lib/regxa_swxa.o \
    /usr/lib/db2.o
```

*Figure 6. Source Code for DATABASE 2 AIX/6000 Switch Load File Makefile*

**Note**

In our installation, we added the highlighted line in Figure 6 to the *db2xa.mk* makefile. Without this we got the following error at region startup:

```
ERZ5801E/0005 03/31/94 13:42:30 sanjose      : Unsuccessful load
of program '/var/cics_regions/sanjose/bin/db2xa'; errno 8.
ERZ5802E/0006 03/31/94 13:42:30 sanjose      : Information on
unsuccessful program load: '3 getgrset /usr/lib/libs.a shr.o'.
ERZ1647E/0232 03/31/94 13:42:30 sanjose      : Abnormal termination
A16D: Unable to load an External Resource Manager XA Support file.
```

This has been fixed with the PTF U426881.

The *make* command creates a loadable file called *db2xa*. You must copy this file to the region directory.

Run the following command:

*cp db2xa /var/cics_regions/<region_name>/bin/db2xa*

where *<region_name>* is the name of your region.

Below we describe the *db2xa* switch; it is for information only. No configuration is required.

Other than the pointers to the *xa_* routines, the fields returned by the DATABASE 2 AIX/6000 *db2xa_switch* are:

| **Field** | **Description** |
|---|---|
| **name** | DB2/6000 |
| **flags** | TMREGISTER\|\|TMNOMIGRATE<br>Explicitly state that the TM should not use association migration or asynchronous operation. Implicitly state that this RM is to use dynamic registration. |
| **version** | value must be 0 |

#### 3.2.2.4 Database Privileges

At region startup the CICS application server makes an initial connection to the RM (in this case DATABASE 2 AIX/6000) using the XA open string. The AIX userid used is *cics*. DATABASE 2 AIX/6000 uses AIX security; therefore the *cics* userid must have the privilege to connect to the database.

This privilege is given by the following commands:

*db2 connect to cicstest*

*db2 grant connect on database to cics*.

### 3.2.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

1. Define XA resources.
2. Set up CICS/6000 region environment.
3. Start up CICS/6000 region.

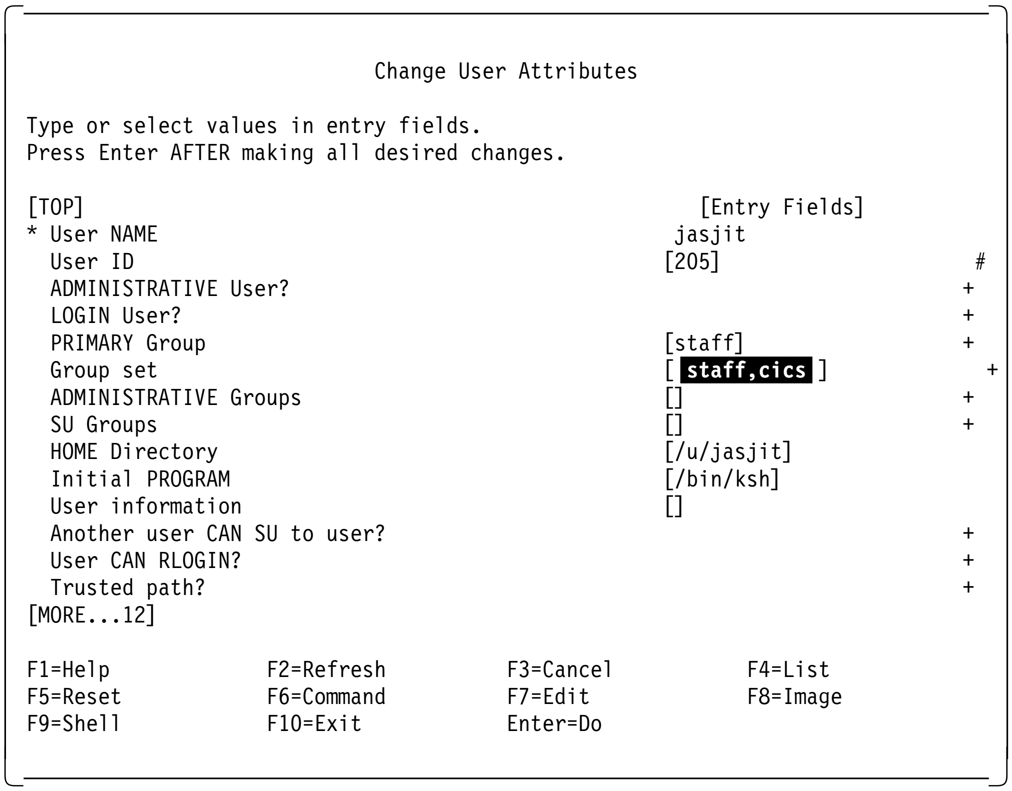
#### 3.2.3.1 XA Resource Definitions

CICS/6000 resources are defined, created, changed, and removed using SMIT. Each resource type has definitions, and these are stored in AIX stanza files.

To access and modify resources in CICS/6000 your AIX userid must belong to the *cics* group. To check whether your userid belongs to this group, you can enter the following command:

```
$ smitty
   └─► Security & Users
      └─► Users
         └─► Change/Show Characteristics of a User
```

or use the fastpath command, *smitty chuser*, to get the Change User Attributes panel shown in Figure 7 on page 21.



```
                          Change User Attributes

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* User NAME                                            jasjit
  User ID                                             [205]              #
  ADMINISTRATIVE User?                                                   +
  LOGIN User?                                                            +
  PRIMARY Group                                       [staff]            +
  Group set                                           [ staff,cics ]       +
  ADMINISTRATIVE Groups                               []                 +
  SU Groups                                           []                 +
  HOME Directory                                      [/u/jasjit]
  Initial PROGRAM                                     [/bin/ksh]
  User information                                    []
  Another user CAN SU to user?                                           +
  User CAN RLOGIN?                                                       +
  Trusted path?                                                          +
[MORE...12]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

*Figure 7. Check That User Is Member of the cics Group*

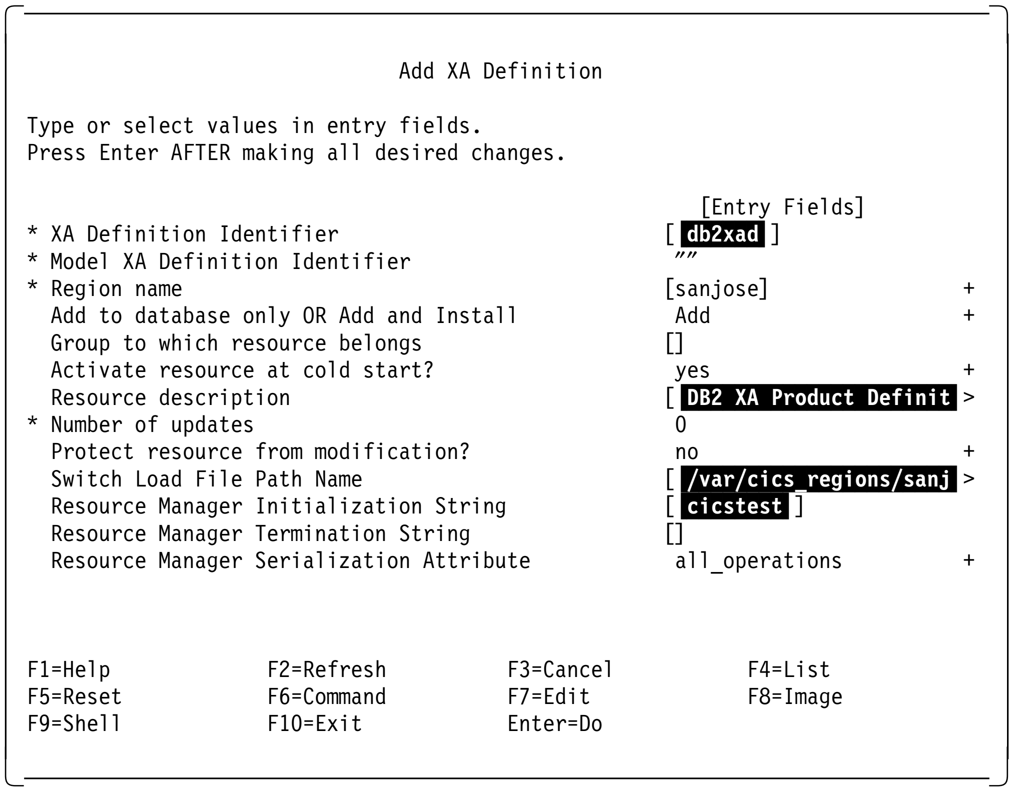
User *jasjit* belongs to the group set containing the groups *cics, staff*, so *jasjit* has the authority to access and modify CICS/6000 resources.

Now, you need to define DATABASE 2 AIX/6000 as a resource manager to CICS/6000.          .

Enter the following command to define a resource manager to the CICS/6000 stanza database:

```
$ smitty cics
  └► Manage CICS/6000 Regions
     └► Define Resources for a CICS/6000 Region
        └► Manage Resource(s)
           └► XA Definitions
              └► Add New
                 └► Model XA Definition Identifier
```

or use the fastpath command, *smitty cicsaddxad*, to get the Add XA Definition panel shown in Figure 8 on page 22.



```
                              Add XA Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* XA Definition Identifier                            [ db2xad ]
* Model XA Definition Identifier                       ""
* Region name                                         [sanjose]            +
  Add to database only OR Add and Install              Add                 +
  Group to which resource belongs                     []
  Activate resource at cold start?                     yes                 +
  Resource description                                [ DB2 XA Product Definit >
* Number of updates                                    0
  Protect resource from modification?                  no                  +
  Switch Load File Path Name                          [ /var/cics_regions/sanj >
  Resource Manager Initialization String              [ cicstest ]
  Resource Manager Termination String                 []
  Resource Manager Serialization Attribute             all_operations      +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 8. DB2 XA Definition for CICS/6000*

The highlighted values are as follows:

- The XA Definition Identifier is unique for every XA definition.
- The Resource description is a comment line.
- The Switch Load File Path name is:

  */var/cics_regions/<region_name>/bin/db2xa*,

  where *<region_name>* is the name of your region. This Switch Load file is the file you created in 3.2.2.3, "Resource Manager Switch" on page 18.
- *cicstest* is the XA open string. No user name or password has been provided. This implies that the database was created with *authentication=client*.

### 3.2.3.2 CICS/6000 Region Environment Setup

In the */var/cics_regions/<region_name>* region directory there is a file named *environment*, which sets up the environment variables on region startup.

The environment variable for DATABASE 2 AIX/6000 is:

| | |
|---|---|
| **DB2INSTANCE** | Defines the instance for CICS/6000 to refer to. |

The syntax for an entry in the environment file is:

**DB2INSTANCE=db2**,

where *db2* is the instance name.

### 3.2.3.3 CICS/6000 Region Startup

Now that you have configured all of the required components, it is time to bring up the CICS/6000 region.

Enter the following command to bring up the CICS/6000 region:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
      └→ Cold Start a CICS/6000 Region
```

or use the fastpath command, *smitty cicscoldstart*, to get the Cold Start CICS/6000 Region panel shown in Figure 9.

```
                          Cold Start CICS/6000 Region

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Region identifier                                  sanjose
  Resource description                              [Region Definition]
  Startup groups                                    []
  Programs to execute at startup                    []
  Programs to execute at phase 1 of shutdown        []
  Programs to execute at phase 2 of shutdown        []
  Name of the default user identifier               [CICSUSER]
  Type of RSL checking for Files                     external              +
  Type of RSL checking for TDQs                      external              +
  Type of RSL checking for TSQs                      external              +
  Type of RSL checking for Journals                  external              +
  Type of RSL checking for Programs                  external              +
  Type of RSL checking for Transactions              external              +
  Do you want to use an External Security Manager?   no                    +
[MORE...54]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

*Figure 9. Cold Start the CICS/6000 Region*

Check the *console.msg* file in the */var/cics_regions/<region_name>* directory for the successful startup of the region. You should see a message similar to this:

```
ERZ8006I/0806 03/31/94 17:50:43 sanjose      : XA_OPEN succeeded:
Application Server 7 connected to 'DB2/6000' using XA_OPEN string
'cicstest'
ERZ1020I/0345 03/31/94 17:50:54 sanjose      : *** CICS/6000 startup
is complete ***
```

## 3.3 Informix Version 5

We divide our discussion of the Informix Version 5 implementation into the following parts:

- Informix Version 5 configuration
- CICS/6000 configuration.

### 3.3.1 Before You Start

Make sure that the following Informix Version 5 products have been installed:

- Informix Online Version 5.01
- Informix TP/XA Version 5.01
- Informix ESQL/C Version 5.01
- Informix ESQL/COBOL Version 5.01.

For more information on the installation of Informix Version 5 products refer to the *Informix UNIX Products Installation Guide Version 5.0.*

### 3.3.2 Informix Version 5 Configuration

The items that you need to consider in the implementation of Informix Version 5 are:

- Shared objects
- XA open string
- Resource manager switch.

We deal with each of these in the sections that follow.

#### 3.3.2.1 Shared Objects

You need to select an Informix Version 5 shared object code. As its name suggests, this shared code is loaded into memory once in the shared library segment and shared by all processes that reference it. The CICS/6000 COBOL run-time environment, CICS/6000 C transactions, and the Switch Load file need to reference the Informix Version 5 shared object at run time.

Two shared objects are defined for the Informix Version 5 TP/XA product.

**libinf501cs.o** In ESQL/COBOL, statement identifiers (created with a PREPARE statement) and cursor names (created with a DECLARE statement) are not case sensitive, by default. If you want the ESQL/COBOL preprocessor to be case sensitive with respect to statement identifiers and cursor names, you should use this object.

**libinf501ci.o** If you are going to be using ESQL/C only or you do not require the ESQL/COBOL preprocessor to be case sensitive with respect to statement identifiers and cursor names, you should use this object.

These objects reside in the */usr/informix/lib/esql* directory.

### 3.3.2.2 XA Open String

The XA open string must have the following syntax:

{*required_fields:*...} [*optional_fields*].

The *required_fields* are:

- ***database_name***

  This is the name of a database that has been created on Informix Version 5. CICS/6000 connects to this database.

The *optional_fields* are:

- **INFORMIXDIR=*dirname***

  This is the full path name of the directory in which the Informix Version 5 products are installed.

  **Note**

  Although the *Informix TP/XA User Manual* states that the INFORMIXDIR=*dirname* field is optional, our experience shows that it is mandatory.

- **TBCONFIG=*tbconfigname***:

  This is the name of the *tbconfig* file used for the *Informix Version 5 Online* RM. If not specified, the default value is *tbconfig*.

- **SQLHOST=*hostname***:

  This is the name of the computer on which the *Informix Version 5 Online RM* resides. If not specified, the default value is the local computer.

- **SQLEXEC=*execname***

  This is an alternative executable name for the *Informix Version 5 Online RM*. This is optional. If it is not specified, the default value is $*INFORMIXDIR/lib/sqlturbo*.

The words in italics are values that you input, and the words in capitals are to be written as is. The string should not contain any blanks or line feeds. Each field is separated by a colon (:).

A sample XA open string for Informix Version 5 is:

*cicstest:INFORMIXDIR=/informix:TBCONFIG=tbconfig.INF*.

**Note**

Informix Version 5 TP/XA does not allow more than one connection to the transaction manager (in this case, CICS/6000). As the XA open string refers to an Informix Version 5 database, a CICS/6000 application can refer to only one database.

#### 3.3.2.3  Resource Manager Switch

Figure 10 on page 27 shows the source code for the Switch Load file for Informix Version 5.  You can find this file in the */usr/lpp/cics/v1.1/src/examples/xa* directory, *informxa.c*.

```
#include <tmxa/xa90.h>

extern struct xa90_switch_t infx_xa_switch;
extern struct xa90_switch_t *cics_xa90_switch;
extern int (*cics_xa_sql_error)();


void cics_xa90_init();

struct sqlca_s
   {
   long sqlcode;
   char sqlerrm[72]
   char sqlerrp[8]
   long sqlerrd[6]
   struct sqlcaw_s
      {
      char sqlwarn0;
      char sqlwarn1;
      char sqlwarn2;
      char sqlwarn3;
      char sqlwarn4;
      char sqlwarn5;
      char sqlwarn6;
      char sqlwarn7;
      } sqlwarn;
   };
extern struct sqlca_s sqlca;
extern long SQLCODE;


extern long *cics_sqlca_code;
extern char cics_sqlca_message[400]

int do_sql_error(void)
{
      char errmsg[400]
      rgetmsg(sqlca.sqlcode, errmsg, sizeof(errmsg));
      cics_sqlca_code = &sqlca.sqlcode;
      strcpy(cics_sqlca_message,errmsg);
}

struct xa_switch_t *CICS_XA_Init(void)
{
   cics_xa_sql_error = &do_sql_error;
   cics_xa90_switch = &infx_xa_switch;
   cics_xa90_init();

   return(&tmxa_xa90Switch);
}
```

*Figure 10. Source Code for Informix Version 5 Switch Load File*

The Switch Load file defines an error handling routine, *do_sql_error*, that allows the display of error messages on the CICS console when set in the sqlca by Informix TP/XA.

To create the Switch Load file *informxa* object you must compile and link the code given in Figure 10 on page 27. Figure 11 shows the source code for the *makefile*. You can also find the code for this makefile in the directory */usr/lpp/cics/v1.1/src/examples/xa* directory, *informxa.mk*.

You must edit the *informxa.mk* file and replace the following:

- **<Full pathname of Informix shared object>** with the appropriate object name. The two objects provided are described in 3.3.2.1, "Shared Objects" on page 24. Only one of these two objects can be used.

  For example, if the object is libinf501cs.o, you must replace the path name with */usr/informix/lib/esql/libinf501cs.o*

- **<Full pathname of Informix lib directory>** with the path name of the directory containing the Informix shared object. For example, if the object is in the /usr/informix/lib/esql directory, the replaced line should be:

  *-L/usr/informix/lib/esql*.

To build the Switch Load file object, *informxa*, issue the following command:

*make -f informxa.mk*.

```
all: informxa.c
    xlc_r -v -I/usr/lpp/encina/include \
    informxa.c \
    -o informxa \
    -eCICS_XA_Init \
    -L <Full pathname of Informix lib directory> \a
    -L/usr/lpp/encina/lib \
    -L/usr/lpp/cics/v1.1/lib \
    -lregxart \
    -lEncServer \
     <Full pathname of Informix share object>
```

*Figure 11. Source Code for Informix Version 5 Switch Load File Makefile*

The *make* command creates a loadable file called *informxa*. You must copy this file to the region directory.

Run the following command:

*cp informxa /var/cics_regions/<region_name>/bin/informxa*,

where *<region_name>* is the name of your region.

Below we describe the *infx_xa* switch; it is for information only. No configuration is required.

Other than the pointers to the *xa_* routines, the fields returned by the Informix Version 5 *infx_xa_switch* are:

| **Field** | **Description** |
|---|---|
| **name** | INFORMIX |
| **flags** | TMNOFLAGS<br>Explicitly state that the TM should not use association migration or asynchronous operation. Also implicitly state that this RM is not to use dynamic registration. |
| **version** | value must be 0. |

## 3.3.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

- Define XA resources.
- Set up CICS/6000 region environment.
- Start up CICS/6000 region.

### 3.3.3.1 XA Resource Definitions

CICS/6000 resources are defined, created, changed, and removed using SMIT. Each resource type has definitions, and these are stored in AIX stanza files.

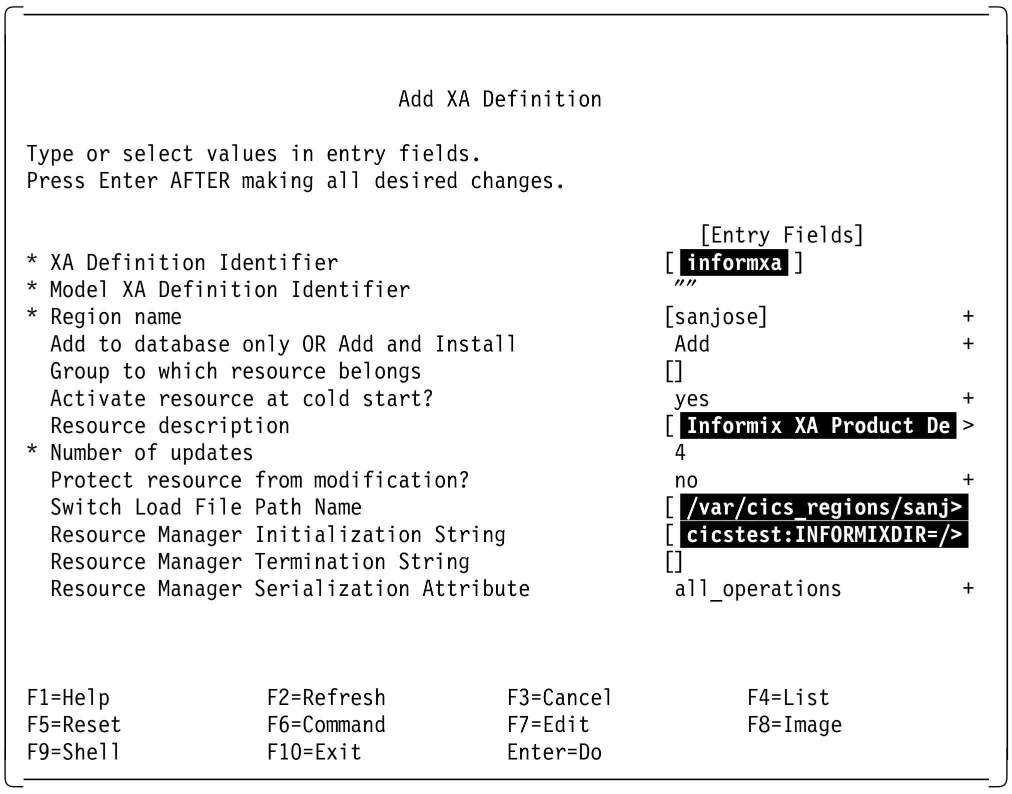
To access and modify resources in CICS/6000 your AIX userid must belong to the **cics** group. To check whether your userid belongs to this group, refer to Figure 7 on page 21.

Now you need to define Informix Version 5 as a resource manager to CICS/6000.

Enter the following command to define a resource manager to the CICS/6000 stanza database:

```
$ smitty cics
  └─► Manage CICS/6000 Regions
      └─► Define Resources for a CICS/6000 Region
          └─► Manage Resource(s)
              └─► XA Definitions
                  └─► Add New
                      └─► Model XA Definition Identifier
```

or use the fastpath command, *smitty cicsaddxad*, to get the Add XA Definition panel shown in Figure 12 on page 30.



```
                             Add XA Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                     [Entry Fields]
* XA Definition Identifier                         [ informxa ]
* Model XA Definition Identifier                    ""
* Region name                                      [sanjose]            +
  Add to database only OR Add and Install           Add                 +
  Group to which resource belongs                  []
  Activate resource at cold start?                  yes                 +
  Resource description                             [ Informix XA Product De >
* Number of updates                                 4
  Protect resource from modification?               no                  +
  Switch Load File Path Name                       [ /var/cics_regions/sanj>
  Resource Manager Initialization String           [ cicstest:INFORMIXDIR=/>
  Resource Manager Termination String              []
  Resource Manager Serialization Attribute          all_operations      +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 12. Informix Version 5 XA Definition for CICS/6000*

The highlighted values are as follows:

- The XA Definition Identifier is unique for every XA definition.
- The Resource description is a comment line.
- The Switch Load File Path Name is:

  */var/cics_regions/<region_name>/bin/informxa*,

  where *<region_name>* is the name of your region. This Switch Load file is the file created in 3.3.2.3, "Resource Manager Switch" on page 26.
- *cicstest:INFORMIXDIR=/informix:TBCONFIG=tbconfig.INF* is the XA open string.

### 3.3.3.2 CICS Region Environment Setup

In every region directory, namely, */var/cics_regions/<region_name>*, there is a file named *environment* which sets up the environment variables on region startup.

There are no required environment variables for Informix Version 5. The variables that can be specified are:

- INFORMIXDIR
- SQLEXEC.

The values of these are as specified in 3.3.2.2, "XA Open String" on page 25.

#### 3.3.3.3 CICS/6000 Region Startup

Now, that you have configured all of the required components, it is time to bring up the CICS/6000 Region.

Enter the following command to bring up the CICS/6000 region:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
      └→ Cold Start a CICS/6000 Region
```

or use the fastpath command: *smitty cicscoldstart*, to get the Cold Start CICS/6000 Region panel shown in Figure 9 on page 23.

Check the *console.msg* file in the */var/cics_regions/<region_name>* directory for the successful startup of the region. You should see a message similar to this:

```
ERZ8006I/0806 03/31/94 17:50:44 sanjose      : XA_OPEN succeeded:
Application Server 7 connected to 'INFORMIX-ONLINE' using XA_OPEN
string 'cicstest:INFORMIXDIR=/home/informix:TBCONFIG=tbconfig.INF'
ERZ1020I/0345 03/31/94 17:50:54 sanjose      : *** CICS/6000 startup
is complete ***
```

## 3.4 Oracle7

We divide our discussion of the Oracle7 implementation into the following parts:

- Oracle7 configuration
- CICS/6000 configuration.

### 3.4.1 Before You Start

Make sure that the following Oracle7 products have been installed:

- ORACLE7 Distributed Database option 7.0.16
- ORACLE7 Server (RDBMS) 7.0.16.4.0
- ORACLE7 XA Library 1.0.3.0.0
- Pro*C 1.5.10.1.0
- Pro*Cobol 1.5.10.1.0
- SQL*Plus 3.1.2.3.1
- SQL*Net V2 2.0.15.0.0
- TCP/IP Protocol Adapter (V2) 2.0.15.0.0.

For more information on the installation of the Oracle7 products refer to *Oracle 7 for IBM RISC/6000 Installation and Configuration Guide*.

## 3.4.2 Oracle7 Configuration

The items that you need to consider in the implementation of Oracle7 are:

1. Shared library
2. XA open string
3. Resource manager switch
4. Table privileges.

We deal with each of these in the sections that follow.

### 3.4.2.1 Shared Library

You need to install an Oracle7 shared library for Oracle7 XA. As its name suggests, the shared code is loaded into memory once in the shared library segment and shared by all processes that reference it. The CICS/6000 COBOL run-time environment, CICS/6000 C transactions, and the Switch Load file need to reference the Oracle7 XA shared library at run time.

During the installation of the Oracle7 XA Library you will be prompted with:

**Oracle XA installation**

Some TP Monitors require a shared version of the ORACLE7 libraries. Do you want to install a shared version of the libraries?

Make sure you select **Yes** to this prompt. The installation will create a shared library in the *$ORACLE_HOME/lib* directory. This library is called *libsharesqlxa.a*. Copy this library to the */usr/lib* directory by running the command as the *root* AIX userid:

*cp $ORACLE_HOME/lib/libsharesqlxa.a /usr/lib*.

### 3.4.2.2 XA Open String

The XA open string must have the following syntax:
*Oracle_XA*{ + *required_fields*...}[ + *optional_fields*]

where the *required_fields* are:

- **Acc=P**{//|*/user/password*}

| | |
|---|---|
| **Acc** | Specifies the user access information |
| **P//** | Indicates that no explicit user or password information is provided and that the *ops$login* form will be used. |
| **P***/user/password:* | Indicates a valid ORACLE userid and the corresponding password. |

- **SesTm=***session_time_limit*

| | |
|---|---|
| **SesTm** | Specifies the maximum amount of time that a transaction can be inactive before the system automatically deletes it. The unit of time is in seconds. |

The *optional_fields* are:

- **DB** = *dbname*

**DB** Specifies the database name

*dbname* Indicates the name Oracle precompilers use to identify the database. This field is required only when applications explicitly specify the database name (that is, use an AT clause in their SQL statements).

- **GPwd=P**/*group_password:*

  **GPwd** Specifies the server security password.

  **P/***group_password* Indicates the server security group password name. Server security groups provide an extra level of protection for different applications running against the same ORACLE instance. The default is an ORACLE-defined server security group.

- **LogDir** = *log_dir*

  **LogDir** Specifies the directory on a local machine where the Oracle XA library error and tracing information can be logged.

  *log_dir* Indicates the path name of the directory. The default is *$ORACLE_HOME/rdbms/log* if *ORACLE_HOME* is set; otherwise it is the current directory.

- **MaxCur** = *maximum_#_of_open_cursors*

  **MaxCur** Specifies the number of cursors to be allocated when the database is opened. It serves the same purpose as the precompiler option, *maxopencursors*.

  *maximum_#_of_open_cursors:* Indicates the number of open cursors to be cached.

- **SqlNet** = *connect_string*

  **SqlNet** Specifies the SQL*Net connect string

  *connect_string:* Indicates the string to log on to the system. The connect string can be either an SQL*Net V1 string, SQL*Net V2 string, or SQL*Net V2 alias. This field is required when you are setting up Oracle on a machine separate from the TP Monitor.

The words in italics are values that you input, and the words in boldface are to be written as is. The string should not contain any blanks or line feeds.

A sample of an XA open string for Oracle7 is:

*Oracle_XA+Acc=P/scott/tiger+SesTm=35+LogDir=/tmp/log.oracle*.

### 3.4.2.3 Resource Manager Switch

Figure 13 on page 34 shows the source code for the Switch Load file for Oracle7. You can find this file in the */usr/lpp/cics/v1.1/src/examples/xa* directory, *oraclexa.c*.

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

extern struct xa_switch_t xaosw;
extern struct xa_switch_t RegXA_xa_switch;
extern struct xa_switch_t *cics_xa_switch;

struct xa_switch_t *CICS_XA_Init(void)
{
   cics_xa_switch = &xaosw;

   cics_xa_init();

   return(&RegXA_xa_switch);
}
```

*Figure 13. Source Code for Oracle7 Switch Load File*

To create the Switch Load file *oraclexa* oracle you must compile and link the code given in Figure 13. Figure 14 on page 35 shows the source code for the *makefile*. You can also find the code for this makefile in the */usr/lpp/cics/v1.1/src/examples/xa* directory, *oraclexa.mk*.

You must edit this *oraclexa.mk* file and replace the following:

- **<Full pathname of Oracle lib directory>** with the path name of the directory containing the Oracle shared library.
- **<Full pathname of Oracle shared library>** with the appropriate library name.

We copied the shared library, *libsharesqlxa.a*, into the ***/usr/lib*** directory. Therefore, the replaced lines should be:

```
-L/usr/lib \
```

```
-lsharesqlxa
```

**Oracle Makefile**

The **oraclexa.mk** makefile may have the home directory and the shared library names already added. Make sure that the home directory name matches the ORACLE_HOME environment variable and that the name of the library is correct.

To build the Switch Load file object, *oraclexa*, issue the following command:

*make -f oraclexa.mk*.

```
all: oraclexa.c
      xlc_r -v -I/usr/lpp/encina/include oraclexa.c \
      -o oraclexa \
      -eCICS_XA_Init \
      -L <Full pathname of Oracle lib directory>  \
      -L/usr/lpp/cics/v1.1/lib \
      -L/usr/lpp/encina/lib \
      -lregxart -lsuperrt -lsupprrt -linfdu -linftrrt -ltasta -ltaslu \
      -l <Full pathname of Oracle shared library>  \
      -lm \
      /usr/lpp/cics/v1.1/lib/regxa_swxa.o
```

*Figure 14. Source Code for Oracle7 Switch Load File Makefile*

The *make* command creates a loadable file called **oraclexa**. You must copy this file to the region directory.

Run the following command:

*cp oraclexa /var/cics_regions/<region_name>/bin/oraclexa*,

where *<region_name>* is the name of your region.

Below we describe the *xaosw* switch; it is for information only. No configuration is required.

Other than the pointers to the *xa_* services, the fields returned by the Oracle7 *xaosw* switch are:

| **Field** | **Description** |
|---|---|
| **name** | ORACLE |
| **flags** | TMNOREGISTER\|\|TMMIGRATE<br>Explicitly state that the TM should not use association migration or aynchronous operation. Also implicitly state that this RM is not to use dynamic registration. |
| **version** | value must be 0. |

### 3.4.2.4 Table Privileges

CICS/6000 connects to the RM with the Oracle userid defined in the XA open string in 3.4.2.2, "XA Open String" on page 32. If the *P//* parameter is specified in the open string, the AIX userid must have the privileges to access the *v$xatrans$* table. Otherwise, the Oracle userid specified in the open string must have the privileges to access the *v$xatrans$* table. This privilege is given by the following command:

*grant select on v$xatrans$ to public*.

## 3.4.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

1. Define XA resources.
2. Set up CICS/6000 region environment.
3. Start up CICS/6000 region.

### 3.4.3.1 XA Resource Definitions

CICS/6000 resources are defined, created, changed, and removed using SMIT. Each resource type has definitions, and these are stored in AIX files in the */var/cics_regions/<region_name>/database* directory.

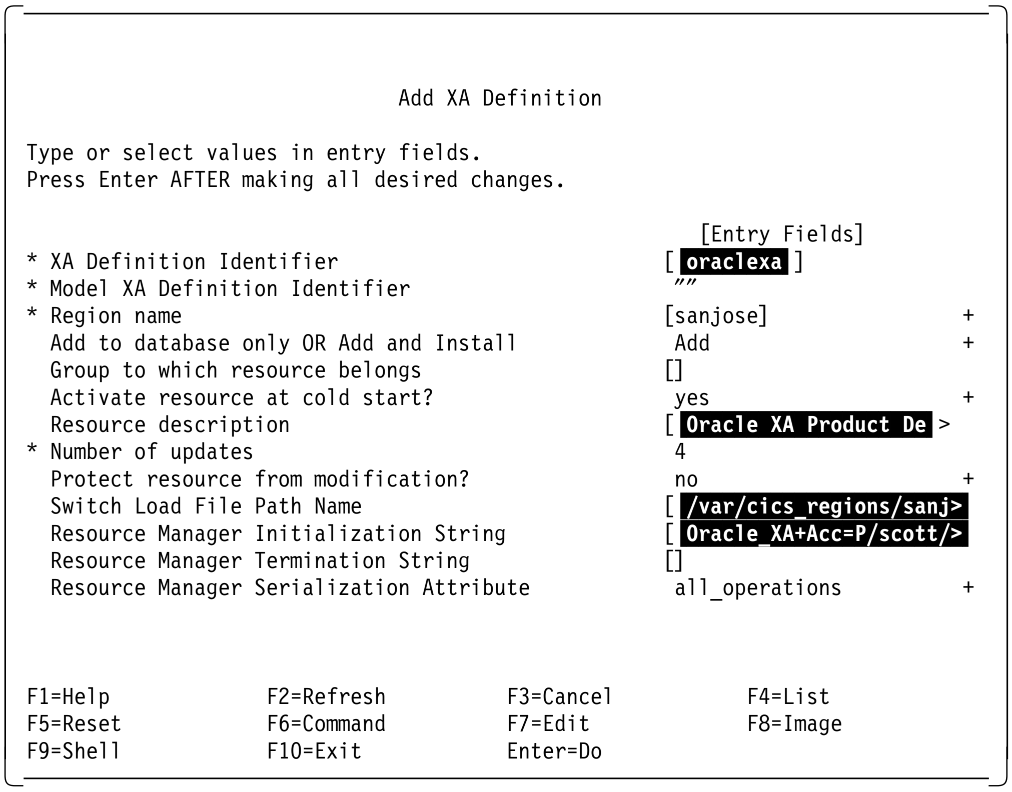
To access and modify resources in CICS/6000 your AIX userid must belong to the *cics* group. To check whether your userid belongs to this group, refer to Figure 7 on page 21.

Now you need to define Oracle7 as a resource manager to CICS/6000.

Enter the following command to define a resource manager to the CICS/6000 stanza database:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
      └→ Define Resources for a CICS/6000 Region
         └→ Manage Resource(s)
            └→ XA Definitions
               └→ Add New
                  └→ Model XA Definition Identifier
```

or use the fastpath command, *smitty cicsaddxad*, to get the Add XA Definition panel shown in Figure 15 on page 37.



```
                              Add XA Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
* XA Definition Identifier                             [ oraclexa ]
* Model XA Definition Identifier                        ""
* Region name                                          [sanjose]              +
  Add to database only OR Add and Install               Add                   +
  Group to which resource belongs                      []
  Activate resource at cold start?                      yes                   +
  Resource description                                 [ Oracle XA Product De >
* Number of updates                                     4
  Protect resource from modification?                   no                    +
  Switch Load File Path Name                           [ /var/cics_regions/sanj>
  Resource Manager Initialization String               [ Oracle_XA+Acc=P/scott/>
  Resource Manager Termination String                  []
  Resource Manager Serialization Attribute              all_operations        +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 15. Oracle7 XA Definition for CICS/6000*

The highlighted values are as follows:

- The XA Definition Identifier is unique for every XA Definition.
- The Resource Description is a comment line.
- The Switch Load File Path Name is:

  */var/cics_regions/<region_name>/bin/oraclexa*,

  where *<region_name>* is the name of your region. This Switch Load file is the file you created in 3.5.2.5, "Resource Manager Switch" on page 42.
- *Oracle_XA+Acc=P/scott/tiger+SesTm=35+LogDir=/tmp/oracle.log* is the XA open string.

### 3.4.3.2 CICS Region Environment Setup

In every region directory, namely, */var/cics_regions/<region_name>*, there is a file named *environment*, which sets up the environment variables on region startup.

Two environment variables are mandatory for Oracle7:

| | |
|---|---|
| **ORACLE_HOME** | Indicates the Oracle home directory |
| **ORACLE_SID** | Indicates the Oracle SID used during the installation of Oracle7 |

#### 3.4.3.3 CICS/6000 Region Startup

Now that you have configured all of the requred components, it is time to bring up the CICS/6000 region.

Enter the following command to bring up the CICS/6000 region: .

```
$ smitty cics
   └─► Manage CICS/6000 Regions
       └─► Cold Start a CICS/6000 Region
```

or use the fastpath command, *smitty cicscoldstart*, to get the Cold Start CICS/6000 Region panel shown in Figure 9 on page 23.

Check the *console.msg* file in the */var/cics_regions/<region_name>* directory for the successful startup of the region. You should see a message similar to this:

```
ERZ8006I/0806 03/31/94 17:50:47 sanjose      : XA_OPEN succeeded: Application
Server 6 connected to ′Oracle_XA′ using XA_OPEN string ′Oracle_XA+Acc=
P/scott/tiger+SesTm=35+LogDir=/tmp′
ERZ1020I/0345 03/31/94 17:50:54 sanjose      : *** CICS/6000 startup is
complete ***
```

## 3.5 Sybase System 10

We divide our discussion of the Sybase System 10 implementation the following parts:

- Sybase System 10 configuration
- CICS/6000 configuration.

### 3.5.1 Before You Start

Make sure that the following Sybase System 10 products have been installed:

- Sybase SQL Server Version 10.0.1
- Sybase Embedded SQL/C
- Sybase Embedded SQL/COBOL
- Open Client/C
- Sybase XA-Library for CICS (we used beta release 10.0.1).

**Note**

The beta version of the XA-Library includes the following:

- XA-Library
- Open Client Client-Library
- CS-Library
- Embedded SQL/C
- Embedded SQL/COBOL.

Installing the beta version of the XA-Library installs all of these dependent products. Subsequent releases of XA-Library will contain only XA-Library files.

For more information on the installation of Sybase System 10 products refer to *Sybase SQL Server Installation Guide for AIX*.

## 3.5.2 Sybase System 10 Configuration

The items that you need to consider in the implementation of Sybase System 10 are:

1. Shared libraries
2. Stored procedures
3. XA configuration file
4. XA open string
5. Resource manager switch
6. Database privileges.

We deal with each of these in the sections that follow.

### 3.5.2.1 Shared Libraries

The shared libraries required for the Sybase System 10 XA-Library are:

- CS-Library
- COMN-Library.

These can be found in the *$SYBASE/lib* directory. They are called *libcomn.so.a* and *libcs.so.a*, respectively. Also check for the existence of the shareable library, *libintl.so.a*.

**Note**

The beta version of the Sybase System 10 XA-Library that we used did not have the *libintl.so.a* library. We also had to create some links for the libraries. In subsequent versions of the Sybase System 10 XA-Library you should not need to do this. The patch for this is given below.

**Beta version shared library patch**

We had to run the following commands to create the shared libraries. (you probably will not need to run them in subsequent releases):

Login as *root*

1. *Create shared library libintl.so.a*
   - cd $SYBASE/lib
   - mkdir tmp
   - cp libintl.a tmp
   - cd tmp
   - touch libintl.shrlist
   - echo ′!#/sybase/lib/libintl.so.a[libintl.so.o]′ >> libintl.shrlist
   - ar w libintl.a >> libintl.shrlist
   - ar x libintl.a
   - cc -bM:SRE -bE:libintl.shrlist -e_nostart -o libintl.so.o *.o
   - ar vr libintl.so.a libintl.so.o
   - cp libintl.so.a $SYBASE/lib
2. *Create library links*
   - cd /
   - mkdir remote
   - mkdir /remote/pokey
   - mkdir /remote/pokey/conn15
   - mkdir /remote/pokey/conn15/csi
   - mkdir /remote/pokey/conn15/csi/build
   - mkdir /remote/pokey/conn15/csi/build/mass
   - mkdir /remote/pokey/conn15/csi/build/mass/distrib
   - mkdir /remote/pokey/conn15/csi/build/mass/distrib/rs6000
   - mkdir /remote/pokey/conn15/csi/build/mass/distrib/rs6000/lib
   - cd /remote/pokey/conn15/csi/build/mass/distrib/rs6000/lib
   - ln -s $SYBASE/lib/libcomn.so.a .
   - ln -s $SYBASE/lib/libintl.so.a .

#### 3.5.2.2 Stored Procedures

To function in a Sybase XA environment for CICS, your Sybase System 10 SQL Server requires additional stored procedures. These procedures are created as follows:

1. Make sure that the server is running by running the following command:

   *ps -ef | grep sybase*,

   where *sybase* is the AIX userid used for the Sybase System 10 installation. You should see a response similar to that in the following screen:

```
  sybase 14123 22058   0 09:07:19  pts/7 37:00 /sybase/bin/dataserver
-d/dev/rmasterlv -sSYBASE -e/sybase/install/errorlog -i/s
  sybase 22058     1   0 09:07:19  pts/7  0:00 sh ./RUN_SYBASE
```

2. Log in as the AIX userid *sybase*.
3. cd $SYBASE
4. isql -Usa -P<*sa_password*> < ./scripts/xa_commit
5. isql -Usa -P<*sa_password*> < ./scripts/xa_sproc.

#### 3.5.2.3 XA Configuration File

The XA configuration file called *xa_config* is located in the *$SYBASE* directory. You must edit this file to reflect your configuration. Each entry in the *xa_config* file has the following format:

[xa]

**lrm** = *lrm_connection*

**server** = *server_name*

**properties** = *prop_list*

**capabilities** = *cap_list*,

where the words in italics are values that you input, and the words in boldface are to be written as is.

The field values are:

- **lrm** = *lrm_connection*
  - The *lrm_connection* is unique to the Sybase System 10 XA environment.
  - The lrm_connection or the logical resource manager (LRM) exist to support multiple connections to a single RM.
  - Each LRM has its own user name and password.

    This is required to control a particular connection′s access to Sybase System 10 SQL Server resources. The user name and password are specified in the XA open string (refer to 3.5.2.4, "XA Open String" on page 42) and as part of the CICS/6000 XA definition (refer to 3.5.3.1, "XA Resource Definitions" on page 45).
- **server** = *server_name*

  The *server_name* is the Sybase System 10 SQL Server to which the TM (CICS/6000 in this case) will connect.
- **properties** = *prop_list*

  Each LRM has preconnection properties. The *prop_list* specifies the preconnection properties to be set. This is an optional field.
- **capabilities** = *cap_list*

  Each LRM has preconnection capabilities. The *cap_list* specifies the preconnection capabilities to be set. This is an optional field.

### 3.5.2.4 XA Open String

The XA open string must have the following syntax:

**-U***username* **-P***password* **-N***connection_name* **-L***log_dir*

where the words in italics are values that you input, and the words in boldface are to be written as is.

The field values are:

- **-U***username*

  Specifies the user access information; *username* indicates a valid Sybase userid.

- **-P***password*

  Indicates the password corresponding to the user.

- **-N***connection_name*

  Indicates a valid LRM name as specified in the *xa_config* file. For details on creating LRMs refer to 3.5.2.3, “XA Configuration File” on page 41.

- **-L***log_dir*

  Indicates the path name of the directory where Sybase XA-Library error information can be logged. This is an optional field. If this field is not specified, error logging is turned off.

A sample of an XA open string for Sybase System 10 is:

*-Usa -Psybase -Nconnection_1 -L/tmp/log.sybase*.

### 3.5.2.5 Resource Manager Switch

Figure 16 on page 43 shows the source code for the Switch Load file for Sybase System 10. CICS/6000 Version 1 Release 1 does not include this file; subsequent releases will include it. However, you can find this file in the Sybase System 10 XA-Library installation. The directory is *$SYBASE/sample/xalibrary/CICS/switch*, and the file is called *sybasexa.c*.

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

extern struct xa_switch_t sybase_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_xa_switch;

   cics_xa_init();

   return(&RegXA_xa_switch);
}
```

*Figure 16. Source Code for Sybase System 10 Switch Load File*

To create the Switch Load file *sybasexa* object you must compile and link the code given in Figure 16 Figure 17 on page 44 shows the source code for the *makefile*. You can also find the code for this makefile in the *$SYBASE/sample/xalibrary/CICS/switch* directory, *sybasexa.mk*.

You must make sure that the *SYBASE* environment variable has been set to the Sybase System 10 installation directory. For example, if the installation directory is */sybase*, the *export SYBASE=/sybase* command sets the SYBASE environment variable.

To build the Switch Load file *sybasexa* object issue the following command:

*make -f sybasexa.mk*.

**Note**

With our configuration we found that we needed to add some more libraries to the makefile. If, after running the *make* command you get some unresolved symbols, add the highlighted libraries and object in Figure 17 on page 44 to your code.

```
SHAROBJS = -lcs.so -lcomn.so
SYB_LIBDIR = $(SYBASE)/lib
SYBLIBS = -lct -lintl -lm -ltcl -linsck -lxa

all: sybasexa.c
      xlc_r -v -I/usr/lpp/encina/include sybasexa.c \
      -o sybasexa \
      -esybasexa \
      -L/usr/lpp/cics/v1.1/lib \
      -L$(SYB_LIBDIR) \
      $(SHAROBJS) \
      $(SYBLIBS) \
      -lregxart \
     -lsupprt -linfdu -linftrrt -lsuperrt -ltasta -ltaslu \
     -l/usr/lpp/cics/v1.1/lib/regxa_swxa.o
```

*Figure 17. Source Code for Sybase System 10 Switch Load File Makefile.*

Below we describe the *sybase_xa* switch; it is for information only. No configuration is required.

Other than the pointers to the *xa_* routines the fields returned by the Sybase System 10 *sybase_xa_switch* are:

| **Field** | **Description** |
|---|---|
| **name** | SYBASE_SQL_SERVER |
| **flags** | TMNOFLAGS<br><br>Explicitly state that the TM should not use association migration or asynchronous operation. Also implicitly state that this RM is not to use dynamic registration. |
| **version** | value must be 0 |

#### 3.5.2.6 Database Privileges

The Sybase System 10 XA open string described in 3.5.2.4, “XA Open String” on page 42 does not explicitly connect to a database. Therefore no database privileges need to be set for the CICS/6000 userid (the userid defined at CICS/6000 installation).

### 3.5.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

1. Define XA resources.
2. Set up CICS/6000 region environment.
3. Start up CICS/6000 region.

### 3.5.3.1 XA Resource Definitions

CICS/6000 resources are defined, created, changed, and removed using SMIT. Each resource type has definitions, and these are stored in AIX stanza files.

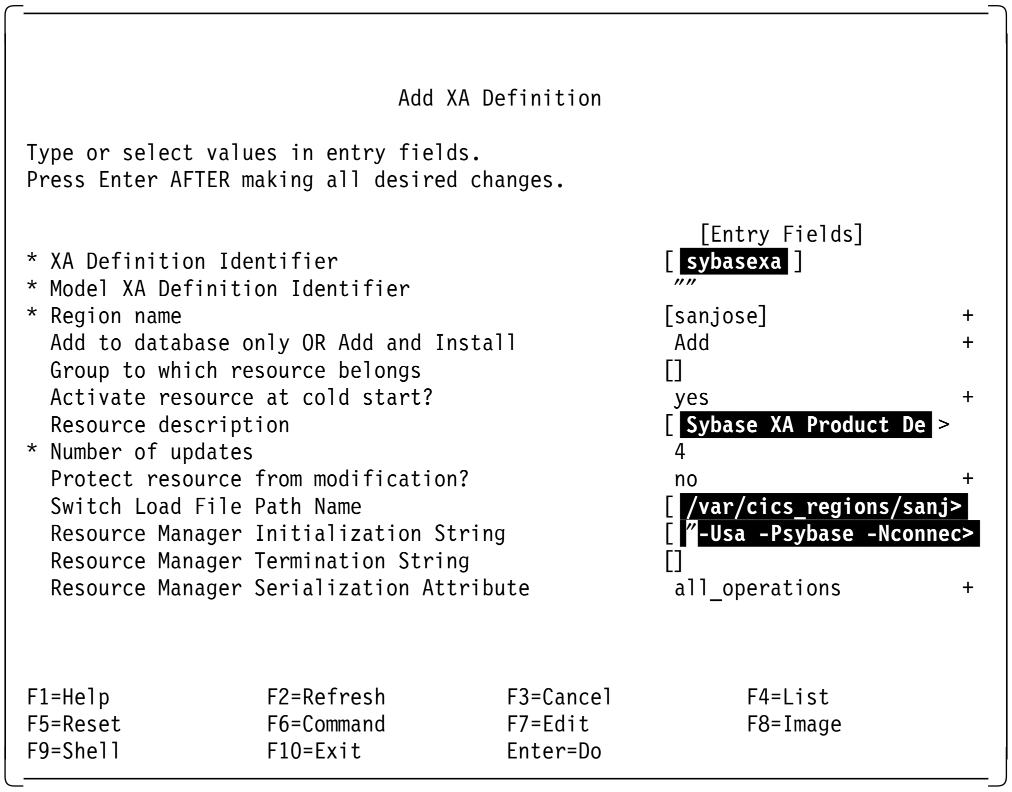
To access and modify resources in CICS/6000 your AIX userid must belong to the *cics* group. To check whether your userid belongs to this group, refer to Figure 7 on page 21.

Now you need to define Sybase System 10 as a resource manager to CICS/6000.

Enter the following command to define a resource manager to the CICS/6000 stanza database:

```
$ smitty cics
  └→ Manage CICS/6000 Regions
     └→ Define Resources for a CICS/6000 Region
        └→ Manage Resource(s)
           └→ XA Definitions
              └→ Add New
                 └→ Model XA Definition Identifier
```

or use the fastpath command, *smitty cicsaddxad*, to get the Add XA Definition panel shown in Figure 18.



```
                              Add XA Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* XA Definition Identifier                           [ sybasexa ]
* Model XA Definition Identifier                      ""
* Region name                                        [sanjose]              +
  Add to database only OR Add and Install             Add                   +
  Group to which resource belongs                    []
  Activate resource at cold start?                    yes                   +
  Resource description                               [ Sybase XA Product De >
* Number of updates                                   4
  Protect resource from modification?                 no                    +
  Switch Load File Path Name                         [ /var/cics_regions/sanj>
  Resource Manager Initialization String             [ "-Usa -Psybase -Nconnec>
  Resource Manager Termination String                []
  Resource Manager Serialization Attribute            all_operations        +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 18. Sybase System 10 XA Definition for CICS/6000*

**Note**

When typing the Resource Manager Initialization String make sure that you begin and end the string with a double quote (″).

The highlighted values are as follows:

- The XA Definition Identifier is unique for every XA Definition.
- The Resource Description is a comment line.
- The Switch Load File Path Name is:

  */var/cics_regions/<region_name>/bin/sybasexa*,

  where *<region_name>* is the name of your region. This Switch Load file is the file you created in 3.5.2.5, "Resource Manager Switch" on page 42.
- *-Usa -Psybase -Nconnection_1 -L/tmp/sybase.log* is the XA open string.

#### 3.5.3.2 CICS Region Environment Setup

In every region directory, namely, */var/cics_regions/<region_name>*, there is a file named *environment*, which sets up the environment variables on region startup.

You need to add a mandatory environment variable for Sybase System 10. The syntax is:

**SYBASE=***install_dir*,

where

*install_dir* is the Sybase System 10 installation directory.

#### 3.5.3.3 CICS/6000 Region Startup

Now that you have configured all of the required components, it is time to bring up the CICS/6000 region.

Enter the following command to bring up the CICS/6000 region.:

```
$ smitty cics
   └─► Manage CICS/6000 Regions
       └─► Cold Start a CICS/6000 Region
```

or use the fastpath command, *smitty cicscoldstart*, to get the Cold Start CICS/6000 Region panel shown in Figure 9 on page 23.

Check the *console.msg* file in the */var/cics_regions/<region_name>* directory for the successful startup of the region. You should see a message similar to this:

```
ERZ8006I/0806 03/31/94 17:50:47 sanjose      : XA_OPEN succeeded: Application
Server 7 connected to 'SYBASE_SQL_SERVER' using XA_OPEN string '-Usa -Psybase
-Nconnection_2 -L/tmp/log.server'
ERZ1020I/0345 03/31/94 17:50:54 sanjose      : *** CICS/6000 startup
is complete ***
```

# Chapter 4. Remote Implementation

In this chapter we describe the implementation steps needed to integrate CICS/6000 with UNIX RDBMSs. We deal with the case where the UNIX RDBMS and the CICS/6000 system reside on separate machines.

We explain the implementation of three UNIX RDBMSs:

- DATABASE 2 AIX/6000
- Oracle7
- Sybase System 10.

**Note**

We do not include Informix Version 5 in this list because it does not support remote access to a CICS/6000 system.

## 4.1 Before You Start

This implementaion is spread over two RISC System/6000s. The UNIX RDBMS client and CICS/6000 reside on one machine. For the purpose of this section we call this the *client* machine. The UNIX RDBMS server resides on the second machine. We will call this the *server* machine.

Check the following on the *client* machine:

- AIX 3.2.5 is installed and running on your RISC System/6000.
- The following CICS/6000 Licensed Program Products (LPPs) are installed:
  - cics6000base.base.obj 1.1.0.0
  - cics6000.dev.obj 1.1.0.0
  - cics6000.prod.obj 1.1.0.0
  - cics6000clt.dev.obj 1.1.0.0
  - cics6000clt.prod.obj 1.1.0.0
  - cics6000mEn_US.msg 1.1.0.0.
- The following Encina LPPs are installed:
  - encServ.obj 1.1.1.0
  - encServmEn_US.msg 1.1.1.0
  - encExec.obj 1.1.1.0
  - encExecmEn_US.msg 1.1.1.0
  - encSfs.obj 1.1.1.0
  - encSfsmEn_US.msg 1.1.1.0.
- The following DCE LPPs are installed:
  - dcebase.base.obj 1.2.0.0
  - dcebase.En_US.msg 1.2.0.0
  - dcebase.admin.obj 1.2.0.0

  - dcebase.appdev.obj 1.2.0.0
  - dcecds.obj 1.2.0.0
  - dcecds.En_US.msg 1.2.0.0
  - dcesec.obj 1.2.0.0
  - dcesec.En_US.msg 1.2.0.0
  - dcepthreads.obj 1.1.0.0
  - dcepthreads.En_US.msg 1.1.0.0.
- The following PTFs have been applied:
  - CICS/6000 December PTFs: U420315, U420263, U422258, U421442
  - DCE base U419616
  - DCE threads U422532.
- DCE is configured on your machine as either a client or a server.
- Encina Structured File Server and logserver are configured and running.
- A CICS/6000 region has been created.
- UNIX RDBMS client.

Check the following on the *server* machine:

- AIX 3.2.5 is installed and running on your RISC System/6000.
- UNIX RDBMS server.

**Note**

We ran the XA interface with CICS/6000 Version 1, Release 1, Level 0. Informix Version 5.02 and Sybase System 10 are officially supported with CICS/6000 Version 1, Release 1, Level 1.

For more information on configuration requirements refer to the *AIX CICS/6000 Planning and Installation Guide*.

## 4.2 DATABASE 2 AIX/6000

We divide our discussion of the DATABASE 2 AIX/6000 implementation into the following parts:

- DATABASE 2 AIX/6000 configuration
- CICS/6000 configuration.

### 4.2.1 Before You Start

Make sure that the following DATABASE 2 AIX/6000 products are installed on the *client*:

- IBM AIX DATABASE 2 Client Application Enabler/6000
- IBM AIX DATABASE2 Software Developer′s Kit.

Make sure that the following DATABASE 2 AIX/6000 products are installed on the *server*:

- IBM DATABASE 2 AIX/6000

- IBM AIX DATABASE 2 Client Support/6000.

Make sure that you have defined an instance on both the DATABASE 2 AIX/6000 *client* and *server* machines.

Creation of an instance is described in 3.2.1, "Before You Start" on page 16.

Make sure that you have created a DATABASE 2 AIX/6000 client-server configuration. The DATABASE 2 AIX/6000 client will reside on the CICS/6000 machine and the DATABASE 2 AIX/6000 server will reside on the remote machine.

Use TCP/IP for the client-server configuration as follows:

1. As root add two new services to the */etc/services* file on both machines. If the service *db21tcpip* does not exist, use two available ports to add:

   ```
   db21tcpip  2130/tcp
   ```

   ```
   dontcare  2131/tcp
   ```

2. DATABASE 2 AIX/6000 *server* setup

   - Log in as instance owner.
   - Register the service name by entering:

     *db2 UPDATE DATABASE MANAGER CONFIGURATION USING SVCNAME db21tcpip*.

   - Create a database available to the client by entering:

     *db2 CREATE DATABASE cicsrem AUTHENTICATION CLIENT*.

   - Set the environment variable **DB2COMM** to **TCPIP** in the *db2profile* file.
   - Start the database with *db2start*.

3. DATABASE 2 AIX/6000 *client* setup

   - Log in as instance owner on the client machine.
   - Catalog the node by entering:

     *db2 CATALOG TCPIP NODE baffin REMOTE baffin SERVER db21tcpip*,

     where *baffin* is the server name.

   - Catalog the database by entering:

     *db2 CATALOG DATABASE cicsrem AS cicsloc AT NODE baffin AUTHENTICATION CLIENT*,

     where:

     *cicsrem* is the database on the DATABASE 2 AIX/6000 server machine, and *cicsloc* is the database alias on the DATABASE 2 AIX/6000 client machine.

   - Connect to the database on the server by entering:

     *db2 CONNECT TO cicsloc*.

### 4.2.2 DATABASE 2 AIX/6000 Configuration

The items that you need to consider in the implementation of DATABASE 2 AIX/6000 are:

- Shared object

  A shared object must be created on the DATABASE 2 AIX/6000 client. Refer to 3.2.2.1, "Shared Object" on page 17 for details on how to create the DATABASE 2 AIX/6000 shared object.

- XA open string

  Refer to 3.2.2.2, "XA Open String" on page 17 for details on the XA open string for DATABASE 2 AIX/6000. The database name in the XA open string must correspond to the alias name that you cataloged on the client machine. For example, in the client setup in 4.2.1, "Before You Start" on page 50 the database alias is *cicsloc*. The XA open string for this is:

  *cicsloc*

- Resource manager switch

  A resource manager switch must be created on the DATABASE 2 AIX/6000 client. Refer to 3.2.2.3, "Resource Manager Switch" on page 18 for details on how to create the resource manager switch for DATABASE 2 AIX/6000.

- Database privileges

  Database privileges must be established on the server machine. Refer to 3.2.2.4, "Database Privileges" on page 20 for details on the database privileges.

### 4.2.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

1. Defina XA resources

   XA resources must be defined on the DATABASE 2 AIX/6000 client. Refer to 3.2.3.1, "XA Resource Definitions" on page 20 for details of these definitions.

2. Set up CICS/6000 region environment

   In every region directory, namely */var/cics_regions/<region_name>*, there is a file named *environment*, which sets up the environment variables on region startup.

   The environment variable for DATABASE 2 AIX/6000 is:

   | | |
   |---|---|
   | **DB2INSTANCE** | Defines the instance for CICS/6000 to refer to. |

   **Multiple DATABASE 2 AIX/6000 Databases**

   If you use two or more DATABASE 2 AIX/6000 databases, whether they are local or remote, you must be careful in setting up your configuration. The **DB2INSTANCE** environment variable can be set only once. Therefore,

   - Locally: You can connect only to databases of the same instance per region.
   - Remotely: You must configure the same instance names on the local and remote machine.

3. Start up CICS/6000 region

   CICS/6000 region startup must be done on the DATABASE 2 AIX/6000 client. Refer to 3.2.3.3, "CICS/6000 Region Startup" on page 23 for information on how to start up the CICS/6000 region.

## 4.3 Oracle7

We divide our discussion of the Oracle7 implementation into the following parts:

- Oracle7 configuration
- CICS/6000 configuration..

### 4.3.1 Before You Start

Make sure that the following Oracle7 products have been installed on the Oracle7 *client* machine:

- ORACLE7 XA Library 1.0.3.0.0
- Pro*C 1.5.10.1.0
- Pro*COBOL 1.5.10.1.0
- SQL*Net V2 2.0.15.0.0
- TCP/IP Protocol Adapter (V2) 2.0.15.0.0.

For more information on the installation of Oracle7 products on the client machine, refer to the *Oracle 7 for IBM RISC/6000 Installation and Configuration Guide*.

Make sure that the following Oracle7 products have been installed on the Oracle7 *server* machine.

- ORACLE7 Distributed Database option 7.0.16
- ORACLE7 Server (RDBMS) 7.0.16.4.0
- ORACLE7 XA Library 1.0.3.0.0
- SQL*Plus 3.1.2.3.1
- SQL*Net V2 2.0.15.0.0
- TCP/IP Protocol Adapter (V2) 2.0.15.0.0.

For more information on the installation of Oracle7 products on the server machine, refer to the *Oracle 7 for IBM RISC/6000 Installation and Configuration Guide*.

Make sure you have created an Oracle7 client-server configuration. The Oracle7 client will reside on the CICS/6000 machine, and the Oracle7 server will reside on the remote machine.

Use TCP/IP for the client-server configuration as follows:

1. Oracle7 *server* setup

   - Check for an entry in the */etc/services* file. There should be an entry similar to this:

```
listener       1522/tcp
```

where listener is the port that the Oracle7 server uses to listen for client requests.

- Create the *listener.ora* file in the */etc* directory. Here is a sample *listener.ora* file:

```
LISTENER =
   (ADDRESS_LIST =
     (ADDRESS =
        (PROTOCOL=IPC)
        (KEY= cics )
     )
     (ADDRESS =
        (PROTOCOL=IPC)
        (KEY= B )
     )
     (ADDRESS =
        (PROTOCOL=TCP)
        (HOST= baffin )
        (PORT= 1522 )
     )
   )
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC=
        (SID_NAME= B )
        (ORACLE_HOME= /oracle )
     )
   )
```

where:

– *cics* is the *service_name* described in the *tnsnames.ora* file.

– *B* is the SID for the Oracle7 installation on the Oracle7 server machine.

– *baffin* is the hostname of the Oracle7 server.

– *1522* is the port number for the listener in the */etc/services* file on the Oracle7 server machine.

– */oracle* is the Oracle7 installation directory.

2. Oracle7 *client* setup

3. Create the *.tnsnames.ora* file in the home directory of the AIX *cics* user. This file allows the client to connect to the Oracle7 server machine. Here is a sample *.tnsnames.ora* file:

```
CICS =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (COMMUNITY = SANJOSE.IBM.COM)
        (PROTOCOL = TCP )
        (HOST = baffin )
        (PORT = 1522 )
      )
    )
    (CONNECT_DATA =
      (SID = B )
    )
  )
```

where:

- *CICS* is the *service_name*. This is a one word descriptor that the client uses to connect to the Oracle7 server.
- *TCP* is the protocol used.
- *baffin* is the hostname of the Oracle7 server.
- *1522* is the port number for the listener in the */etc/services* file on the Oracle7 server machine.
- *B* is the SID for the Oracle7 installation on the Oracle7 server machine.

4. Check the client-server configuration from the Oracle7 client machine:

   a. log in as the AIX user *cics*

   b. *export ORACLE_HOME=/oracle*,

      where */oracle* is the Oracle7 installation directory.

   c. *export PATH=$PATH:$ORACLE_HOME/bin*

   d. *sqlplus scott/tiger@cics*,

      where *cics* is the service_name as specified in the *.tnsnames.ora* file.

## 4.3.2 Oracle7 Configuration

The items that you have to consider in the implementation of Oracle7 are:

- Shared library

  During the installation of the Oracle7 XA Library you will be prompted with:

  **Oracle XA Installation**

  Some TP monitors require a shared version of the ORACLE7 libraries.
  Do you want to install a shared version of the libraries?

  Make sure you select **Yes** to this prompt. The installation will create a shared library in the *$ORACLE_HOME/lib* directory. This library is called *libsharesqlxa.a*. Copy this library to the */usr/lib* directory by running the command as the *root* AIX userid:

  *cp $ORACLE_HOME/lib/libsharesqlxa.a /usr/lib*

- XA open string

  The syntax of the XA open string is described in 3.4.2.2, "XA Open String" on page 32. You will need to add an extra field to the XA open string. This field allows the CICS/6000 region to access data on the remote Oracle7 server. The extra field is:

**SqlNet** = *connect_string*

| | |
|---|---|
| **SqlNet** | Specifies the SQL*Net connect string |
| *connect_string:* | Indicates the string to log on to the system. The connect string can be either an SQL*Net V1 string, SQL*Net V2 string, or SQL*Net V2 alias. This string is required when you are setting up Oracle on a machine separate from the TP monitor. |

For example, if you were to use the *.tnsnames.ora* and *listener.ora* files listed in 4.3.1, “Before You Start” on page 53, the XA open string would be:

*Oracle_XA+Acc=P/scott/tiger+SesTm=35+LogDir=/tmp/xa.log+SqlNet=cics*

where *cics* is the service_name as specified in the *.tnsnames.ora* file.

- Resource manager switch

  You need to create the resource manager switch on the Oracle7 client machine. Refer to 3.4.2.3, “Resource Manager Switch” on page 33 for details on how to create the resource manager switch.

- Table privileges

  Table privileges must be established on the Oracle7 server machine. For details refer to 3.4.2.4, “Table Privileges” on page 35.

## 4.3.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

1. Define XA resources

   XA resources must be defined on the Oracle7 client machine. For details refer to 3.4.3.1, “XA Resource Definitions” on page 36.

2. Set up CICS/6000 region environment

   In every region directory, namely */var/cics_regions/<region_name>*, there is a file named *environment*, which sets up the environment variables on region startup.

   There are no mandatory environment variables for Oracle7 in the client-server setup. However, there is an optional variable:

   | | |
   |---|---|
   | **TNS_ADMIN** | Indicates the directory where the *tnsnames.ora* file can be located. If this variable is not set, the *.tnsnames.ora* file must exist in the AIX *cics* userid. |

   **Note**

   There is a dot ( **.** ) prefix on the tnsnames.ora file when the TNS_ADMIN variable is not set.

3. Start up CICS/6000 region

   CICS/6000 region startup must be done on the Oracle7 client machine. For details refer to 3.4.3.3, “CICS/6000 Region Startup” on page 38.

## 4.4 Sybase System 10

We divide our discussion of the Sybase System 10 implementation into the following parts:

- Sybase System 10 configuration
- CICS/6000 configuration.

### 4.4.1 Before You Start

Make sure that the following Sybase System 10 products have been installed on the same machine as CICS/6000, that is, the Sybase System 10 *client* machine:

- Sybase Embedded SQL/C
- Sybase Embedded SQL/COBOL
- Open Client/C
- Sybase XA-Library for CICS (we used beta release 10.0.1).

For more information on the installation of Sybase System 10 products refer to the *Sybase SQL Server Installation Guide for AIX*.

Make sure that the following Sybase System 10 products have been installed on the *server* machine:

- Sybase SQL Server Version 10.0.1
- Sybase XA-Library for CICS (we used beta release 10.0.1)

**Note**

The beta version of the XA-Library includes the following:

- XA-Library
- Open Client Client-Library
- CS-Library
- Embedded SQL/C
- Embedded SQL/COBOL.

Installing the beta version of the XA-Library installs all of these dependent products. Subsequent releases of XA-Library will contain only XA-Library files.

Make sure you have created a Sybase System 10 client-server configuration. The Sybase System 10 client will reside on the CICS/6000 machine, and the Sybase System 10 server will reside on the remote machine.

Use TCP/IP for the client-server configuration as follows:

1. Sybase System 10 *server* setup
   - Check for an entry in the */etc/services* file. There should be an entry similar to this:

```
SYBASE query    2025/tcp
```

where SYBASE is the Sybase System 10 SQL Server on the remote machine.

- The server requires the *interfaces* file in the *$SYBASE* directory to listen for clients.
- Use *sybinit* to create the *interfaces* file entry. For details refer to *Sybase SQL Server Installation Guide for AIX.* This entry should match the entry in the */etc/services* file. For example, the interfaces file entry corresponding to the */etc/services* above is:

```
SYBASE
        query tcp ether bengal 2025
        master tcp ether bengal 2025
```

  where *bengal* is the hostname of the Sybase System 10 SQL Server.

- Set the environment variable *DSLISTEN* to the Sybase System 10 SQL Server name. To do this run the following command:

  ***export DSLISTEN=****server_name*,

  where *server_name* is the name of the Sybase System 10 SQL Server specified at installation time. If *DSLISTEN* is not set, the default value is *SYBASE*.

2. Sybase System 10 *client* setup

   - Set the server name

     The environment variable for this is *DSQUERY*. To set this variable run the following command:

     ***export DSQUERY=****server_name*,

     where *server_name* is the name of the Sybase System 10 SQL Server on the remote machine.

   - Create an entry in the *interfaces* file

     Use *sybinit* to create an entry. This entry should be the same as the entry in the *interfaces* file for the remote Sybase System 10 SQL Server. For example, the entry should be similar to this:

```
SYBASE
         query tcp ether bengal 2025
         master tcp ether bengal 2025
```

     where *SYBASE* is the Sybase System 10 SQL Server name on the remote server machine.

3. Check the client-server configuration from the Sybase System 10 client machine.

   a. log in as the AIX user *cics*.

   b. *export SYBASE=/sybase*,

      where */sybase* is the Sybase System 10 installation directory.

   c. *export PATH=$PATH:$SYBASE/bin*

   d. *export DSQUERY=SYBASE*,

where *SYBASE* is the Sybase System 10 SQL Server name.

e. *isql -Usa -Psybase*,

where *sybase* is the password for the user *sa*.

### 4.4.2 Sybase System 10 Configuration

The items you need to consider in the implementation of Sybase System 10 are:

- Shared libraries

  You must create the shared libraries on the Sybase System 10 client machine. Refer to 3.5.2.1, "Shared Libraries" on page 39 for details on how to create the shared libraries.

- Stored procedures

  You must run the stored procedures on the Sybase System 10 server machine. To function in a Sybase XA environment for CICS, your SQL server requires additional stored procedures. Refer to 3.5.2.2, "Stored Procedures" on page 40 for details on how to create the stored procedures.

- XA configuration file

  You must create an XA configuration file on the Sybase System 10 client machine. The XA configuration file called *xa_config* is located in the *$SYBASE* directory. Refer to 3.5.2.3, "XA Configuration File" on page 41 for details on how to add an entry to this file.

- XA open string

  For details on the XA open string for Sybase System 10 refer to 3.5.2.4, "XA Open String" on page 42.

- Resource manager switch

  You must create the resource manager switch on the Sybase System 10 client machine. For details on the creation of the resource manager switch refer to 3.5.2.5, "Resource Manager Switch" on page 42.

- Database privileges

  The Sybase System 10 XA open string described in 3.5.2.4, "XA Open String" on page 42 does not explicitly connect to a database. Therefore no database privileges need to be set for the CICS/6000 userid (the userid defined when the CICS/6000 was installed).

### 4.4.3 CICS/6000 Configuration

The steps required for the CICS/6000 setup are as follows:

1. Define XA resources

   XA resources must be defined on the Sybase System 10 client machine. For details refer to 3.5.3.1, "XA Resource Definitions" on page 45.

2. Set up CICS/6000 region environment

   In every region directory, namely */var/cics_regions/<region_name>*, there is a file named *environment*, which sets up the environment variables on region startup.

   You need to add the following environment variable for the Sybase System 10 client-server configuration:

   **SYBASE=***sybase_dir*,

where *sybase_dir* is the Sybase System 10 installation directory.

**Note**

If you set the environment variable *DSQUERY* in the region *environment* file, it has no effect on the Sybase System 10 SQL Server to which the CICS/6000 region will connect. This is unlike the Sybase System 10 client-server configuration without a TP monitor, where setting the *DSQUERY* variable defines the Sybase System 10 SQL Server. In a CICS/6000 environment the LRM connection name is used to check the corresponding Sybase System 10 SQL Server name from the *xa_config* file.

3. Start up CICS/6000 region

   CICS/6000 region startup must be done on the Sybase System 10 client machine. For details refer to 3.5.3.3, "CICS/6000 Region Startup" on page 46.

# Chapter 5. Resource Manager Initialization at Region Startup: A Diagrammatic View

In this chapter we review the steps required to start up a CICS/6000 region when connecting to an XA-compliant RDBMS. We also show the various AIX processes created for each RDBMS whenever a CICS/6000 region is started up. We discuss the following four databases:

- DATABASE 2 AIX/6000
- Informix Version 5
- Oracle7
- Sybase System 10.

## 5.1 DATABASE 2 AIX/6000

In this section we describe the process flow at DATABASE 2 AIX/6000 registration and the AIX processes generated.

### 5.1.1 Registration

Figure 19 on page 62 depicts the DATABASE 2 AIX/6000 local and remote registration process.

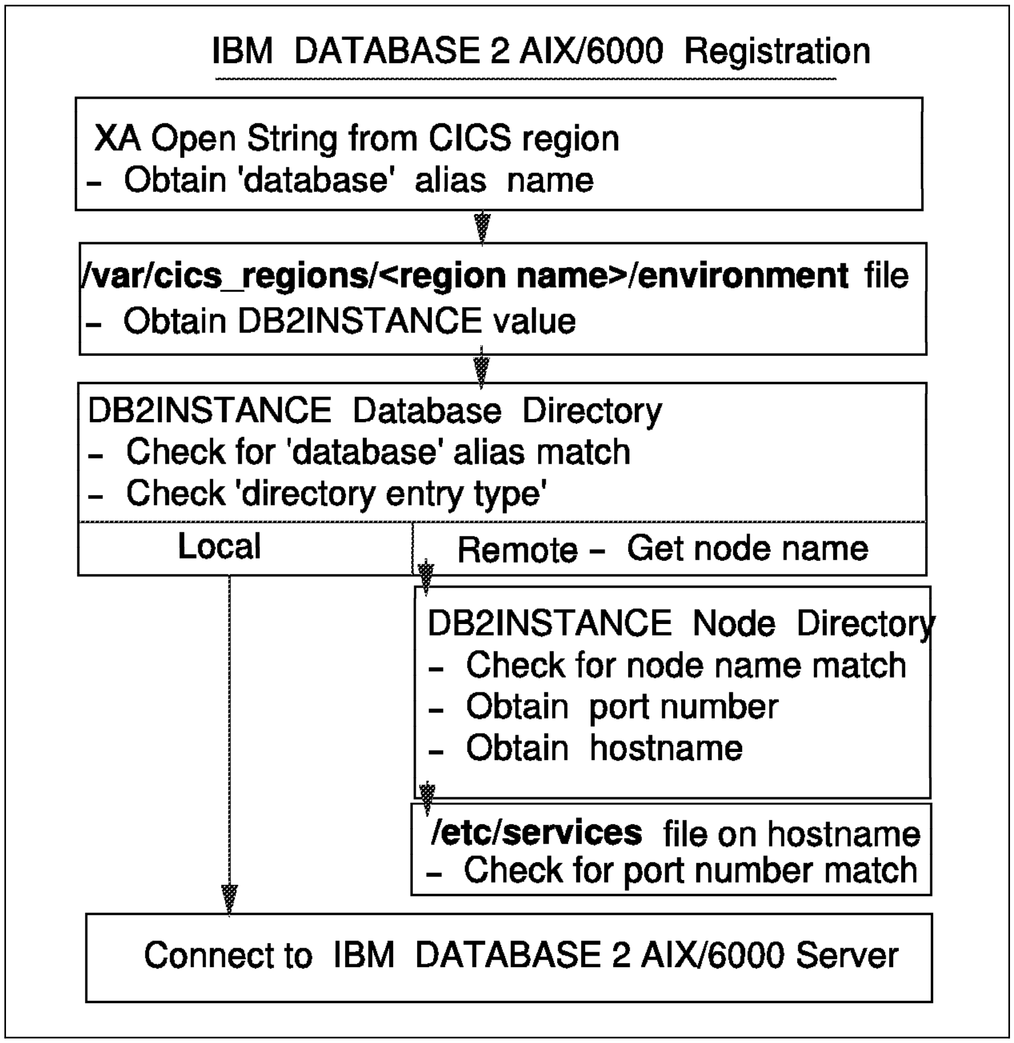

IBM DATABASE 2 AIX/6000 Registration
XA Open String from CICS region
- Obtain 'database' alias name
/var/cics_regions/<region name>/environment file
- Obtain DB2INSTANCE value
DB2INSTANCE Database Directory
- Check for 'database' alias match
- Check 'directory entry type'
Local
Remote - Get node name
DB2INSTANCE Node Directory
- Check for node name match
- Obtain port number
- Obtain hostname
/etc/services file on hostname
- Check for port number match
Connect to IBM DATABASE 2 AIX/6000 Server

*Figure 19. DATABASE 2 AIX/6000 Registration: Local and Remote (Client-Server)*

The database alias is obtained from the XA open string. The region environment file is checked to obtain the *DB2INSTANCE* value. Then the database directory of the *DB2INSTANCE* is checked for a match in the database alias name. If the implementation is local, a connection is established to the DATABASE 2 AIX/6000 server. If the implementation is remote, the node name is obtained from the database directory and is checked against the node name in the node directory. Once a match is found, the port number and the hostname are obtained and checked against the */etc/services* file of the hostname for a port number match. If those port numbers match, a connection is established to the DATABASE 2 AIX/6000 server on the hostname system.

## 5.1.2 AIX Processes

When you have started the region successfully, issue the following command to see the AIX processes for DATABASE 2 AIX/6000:

*ps -ef | grep db2*

For the local DATABASE 2 AIX/6000 implementation you will see:

```
    db2  5858 23066   0 17:25:41      -  0:00  db2agent
    db2  9451 10779   0 17:25:57      -  0:00 db2dlock
    db2 10779 23066   0   Mar 25      -  0:00 db2dlock
   root 12527 17188   5 17:27:52  pts/9  0:00 grep db2
    db2 13276 23066   0 17:25:38      -  0:00  db2agent
    db2 14566 23066   0 17:25:43      -  0:00  db2agent
   root 18713     1   0   Mar 25      -  0:00 db2wdog
    db2 23066 18713   0   Mar 25      -  1:36 db2sysc
    db2 25030 10368   0   Mar 23  pts/6  0:01 -ksh
   root 28312     1   0 17:22:09      -  0:00 db2licd
    db2 28700 23066   0   Mar 25      -  0:00 [db2loggr]
    db2 32234 28700   0 17:25:57      -  0:00 [db2loggr]
```

The highlighted processes service the CICS/6000 application servers. There is one *db2agent* process for each CICS/6000 application server.

For the remote DATABASE 2 AIX/6000 implementation you will see:

```
    db2  4864 22783   0 10:52:08      -  0:00 db2dlock
    db2 18434 22783   0 10:52:09      -  0:00  db2tcpcm
   root 20478     1   0 10:52:08      -  0:00 db2wdog
    db2 22105 28417   0 11:14:51      -  0:00 [db2loggr]
    db2 22783 20478   0 10:52:08      -  0:00 db2sysc
    db2 26458 18434   0 11:14:51      -  0:00  db2agent
    db2 27749 29684  13 11:15:51  pts/7  0:00 ps -ef
    db2 28417 22783   0 10:52:08      -  0:00 [db2loggr]
    db2 29684 31475   0 10:51:36  pts/7  0:00 -ksh
    db2 29788  4864   0 11:14:52      -  0:00 db2dlock
    db2 29955 22783   0 10:52:10      -  0:00  db2tcpim
    db2 30299 18434   0 11:14:52      -  0:00  db2agent
    db2 30822 29684   1 11:15:51  pts/7  0:00 grep db2
   root 32016     1   0 10:57:49      -  0:00 db2licd
    db2 32344 18434   0 11:14:51      -  0:00  db2agent
```

The *db2agent* processes service the CICS/6000 application servers. In addition, the *db2tcpim* and *db2tcpcm* processes are required for the client-server configuration. There is one *db2agent* process for each CICS/6000 application server.

**Note**

When we ran two DATABASE 2 AIX/6000 servers—one on the local machine and one on the server machine—there were three *db2agent* processes on the local machine but none on the remote machine.

## 5.2 Informix Version 5

In this section we describe the process flow at Informix Version 5 registration and the AIX processes generated.

### 5.2.1 Registration

Figure 20 depicts the Informix Version 5 local registration process.

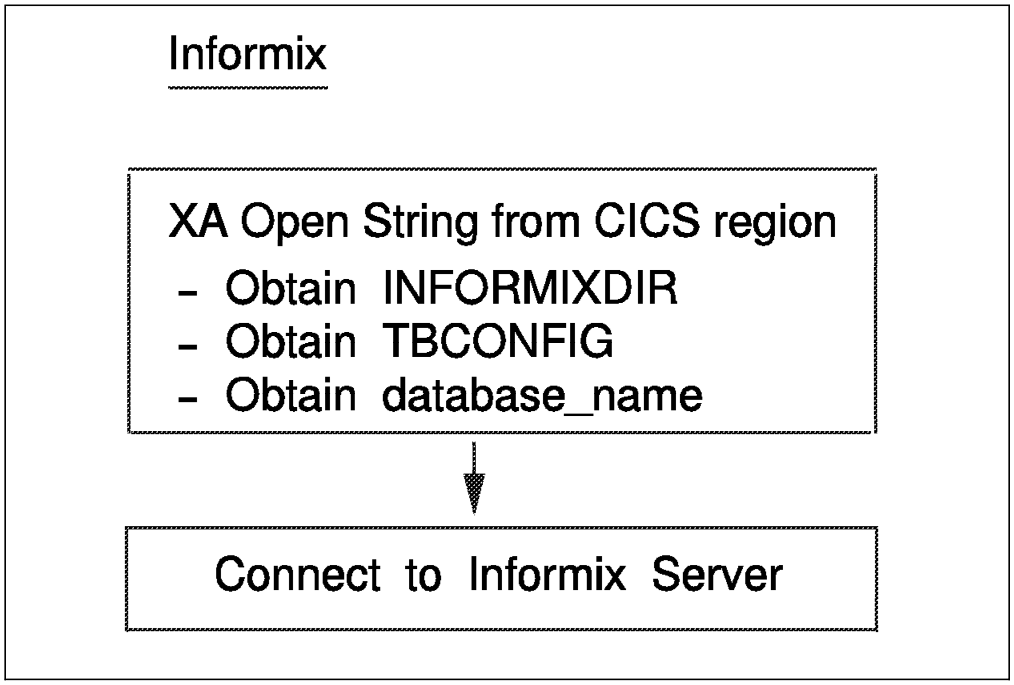


*Figure 20. Informix Version 5 Local Registration*

The *database_name*, *INFORMIXDIR*, and *TBCONFIG* are obtained from the XA open string, and, with these values, a connection is established to the Informix Version 5 server.

### 5.2.2 AIX Processes

When you have started the region successfully, issue the following commands to see the AIX processes for Informix Version 5:

*ps -ef | grep informix*

and

*ps -ef | grep sqlturbo*

For the Informix Version 5 implementation you will see:

```
informix 11555 28472   0   Mar 29      -  0:00 tbundo
    root 12529 17188   7 17:28:14  pts/9  0:00 grep informix
informix 25473 21786   0   Mar 23  pts/6  0:01 -ksh
informix 28472     1   0   Mar 25      - 10:33 tbinit
informix 31545 28472   0   Mar 25      -  0:15 tbpgcl
    root 12536 17188   5 17:29:04  pts/9  0:00 grep sqlturbo
    cics 21479 23515   0 17:25:43      -  0:00  sqlturbo cics 5.01.U
     RDS#N000000 -x cicstest
    cics 26339 12249   0 17:25:41      -  0:00  sqlturbo cics 5.01.U
     RDS#N000000 -x cicstest
    cics 33247 13786   0 17:25:40      -  0:00  sqlturbo cics 5.01.U
     RDS#N000000 -x cicstest
```

The highlighted processes service the CICS/6000 application servers. There is one *sqlturbo* process for each CICS/6000 application server.

## 5.3 Oracle7

In this section we describe the process flow at Oracle7 registration and the AIX processes generated.

### 5.3.1 Registration

Figure 21 depicts the Oracle7 local registration process.

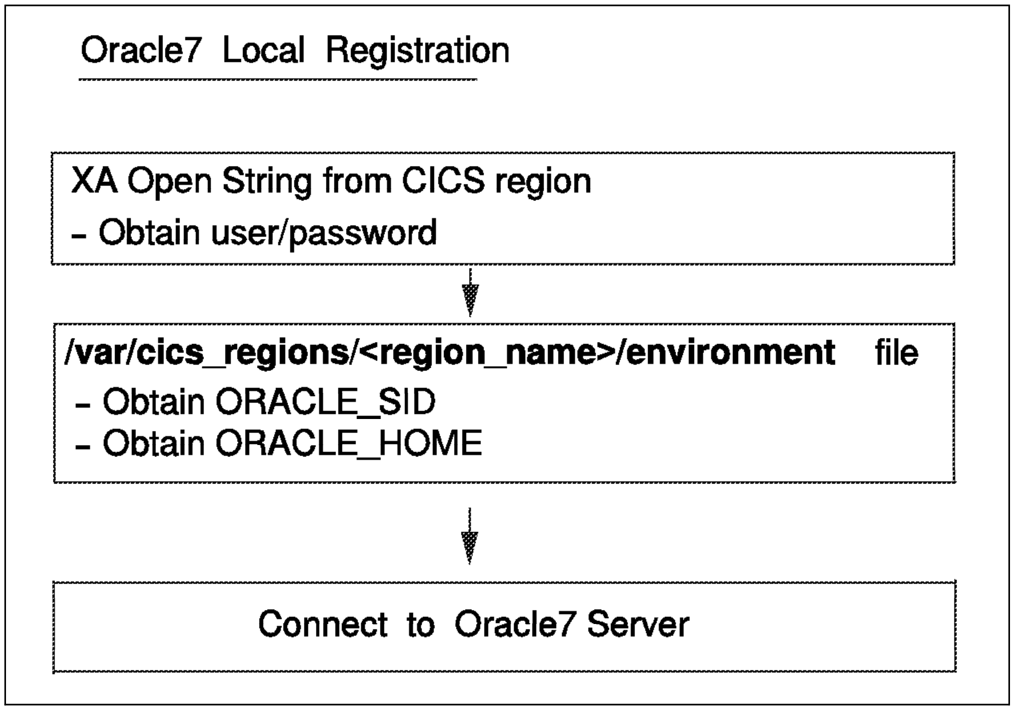

Oracle7 Local Registration
XA Open String from CICS region
- Obtain user/password
/var/cics_regions/<region_name>/environment file
- Obtain ORACLE_SID
- Obtain ORACLE_HOME
Connect to Oracle7 Server

*Figure 21. Oracle7 Local Registration*

The user and password are obtained from the XA open string. The Oracle7 environment variable values, *ORACLE_SID* and *ORACLE_HOME*, are obtained from the region environment file. With these values a connection is established to the Oracle7 server.

Figure 22 on page 66 depicts the Oracle7 remote registration process.

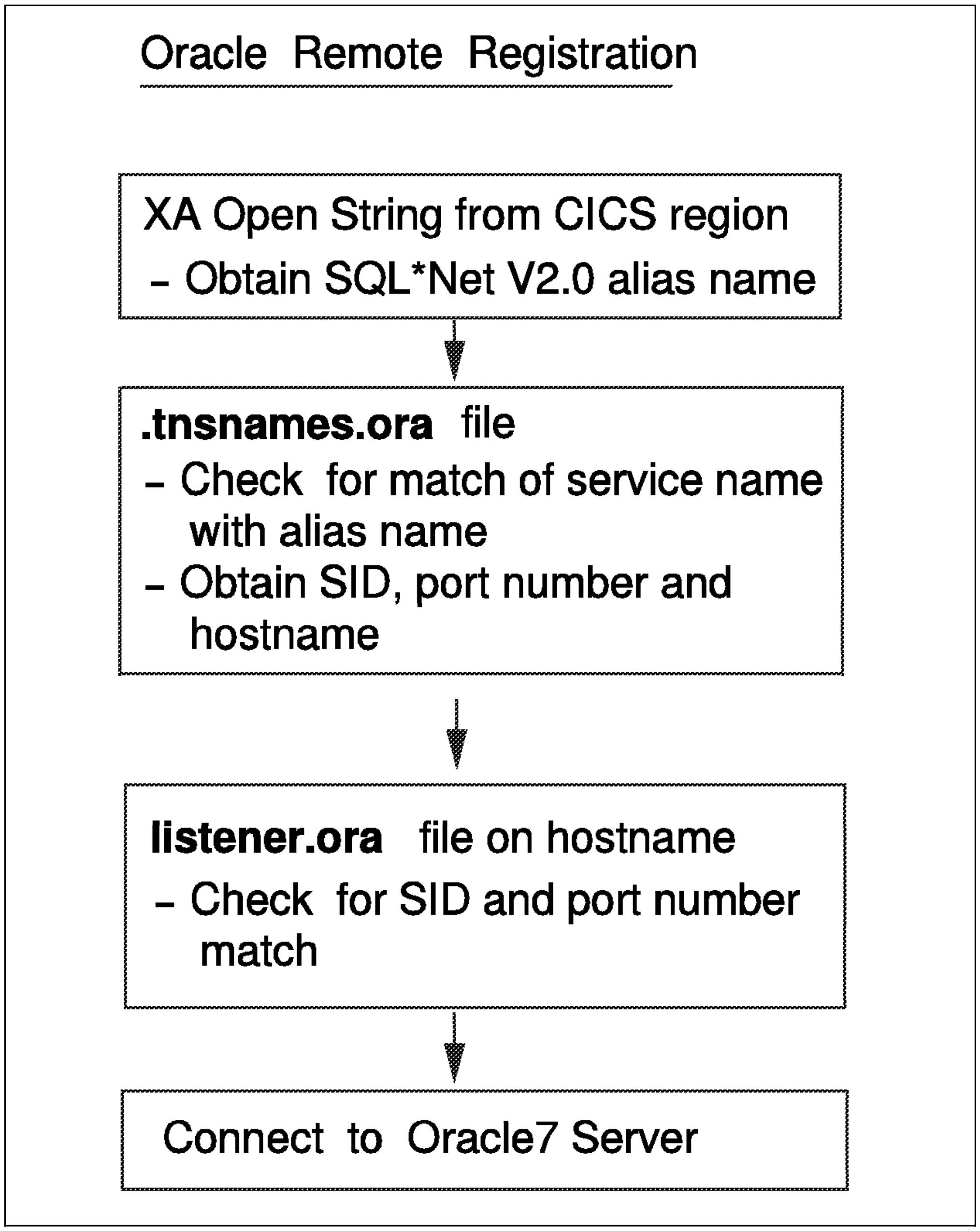

Oracle Remote Registration
XA Open String from CICS region
- Obtain SQL*Net V2.0 alias name
.tnsnames.ora file
- Check for match of service name with alias name
- Obtain SID, port number and hostname
listener.ora file on hostname
- Check for SID and port number match
Connect to Oracle7 Server

*Figure 22. Oracle7 Remote Registration (Client-Server)*

The SQL*Net V2.0 alias name is obtained from the XA open string. A match for this is to be found in the *.tnsnames.ora* file. Subsequently the Oracle7 SID value, port number, and hostname are obtained from this file. The *listener.ora* file on the hostname is checked for a match in the port number and Oracle7 SID value. On the success of this match a connection on the hostname machine is established to the Oracle7 server.

### 5.3.2 AIX Processes

When you have started the region successfully, issue the following command to see the AIX processes for Oracle7:

```
ps -ef | grep oracle
```

For a local Oracle7 implementation you will see:

```
     cics  8677     1   0 17:25:42      -  0:00  oracleA (DESCRIPTION=
      (LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
   oracle 10642     1   0   Mar 29      -  0:02 ora_reco_A
     root 12540 17188   3 17:29:22  pts/9  0:00 grep oracle
     cics 13033     1   0 17:25:44      -  0:00  oracleA (DESCRIPTION=
      (LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
   oracle 19557 27763   0   Apr 03  pts/2  0:00 -ksh
   oracle 20110     1   0   Mar 29      -  2:32 ora_pmon_A
   oracle 20879     1   0   Mar 29      -  5:16 ora_dbwr_A
   oracle 21136     1   0   Mar 29      -  2:38 ora_lgwr_A
   oracle 24747     1   0   Mar 29  pts/0  0:00 /oracle/bin/tnslsnr
     LISTENER -inherit
   oracle 31377     1   0   Mar 29      -  1:34 ora_smon_A
     cics 33505     1   0 17:25:40      -  0:00  oracleA (DESCRIPTION=
      (LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
```

The highlighted processes service the CICS/6000 application servers. There is one *oracleA* process for each CICS/6000 application server.

For a remote Oracle7 implementation you will see:

```
     root  6924 29948   3 17:31:44  pts/1  0:00 grep oracle
   oracle 18831     1   0   Mar 29      -  1:00 ora_smon_B
   oracle 19597     1   0   Mar 29      -  4:07 ora_dbwr_B
   oracle 21491     1   0 17:25:45      -  0:00  oracleB (LOCAL=NO)
   oracle 23182     1   0   Mar 29      -  2:06 ora_lgwr_B
   oracle 23440     1   0   Mar 29      -  0:02 ora_reco_B
   oracle 23948     1   0   Mar 29      -  1:54 ora_pmon_B
   oracle 25316     1   0 14:32:06      -  0:00 /oracle/bin/tnslsnr
     LISTENER -inherit
   oracle 30449     1   0 17:25:45      -  0:00  oracleB (LOCAL=NO)
   oracle 30959     1   0 17:25:43      -  0:00  oracleB (LOCAL=NO)
```

The highlighted processes service the CICS/6000 application servers. There is one *oracleB* process for each CICS/6000 application server.

**Note**

We used the *dedicated server configuration* for Oracle7 where a server handles requests for a single-user process. In an Oracle7 *multithreaded server configuration*, many user processes share a small number of server processes. Using the multithreaded server reduces the number of Oracle7 AIX processes.

## 5.4 Sybase System 10

In this section we describe the process flow at Sybase System 10 registration and the AIX processes generated.

## 5.4.1 Registration

Figure 23 depicts the Sybase System 10 local and remote registration process.

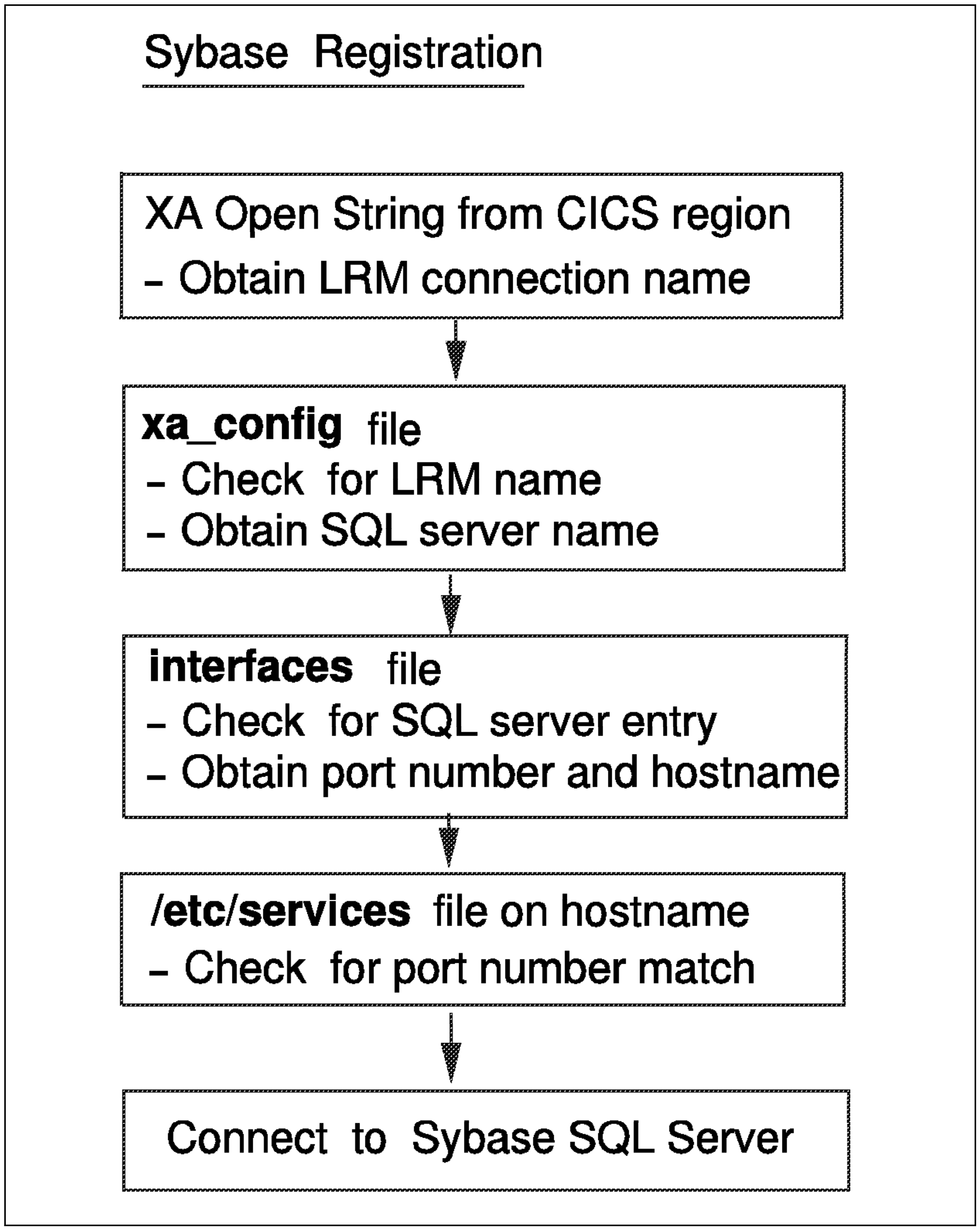

Sybase Registration
XA Open String from CICS region
- Obtain LRM connection name
xa_config file
- Check for LRM name
- Obtain SQL server name
interfaces file
- Check for SQL server entry
- Obtain port number and hostname
/etc/services file on hostname
- Check for port number match
Connect to Sybase SQL Server

*Figure 23. Sybase System 10 Registration: Local and Remote (Client-Server)*

The LRM connection name is obtained from the XA open string. A match for this name is obtained from the *xa_config* file, and the Sybase System 10 SQL server name corresponding to the LRM name is obtained. After this. The *interfaces* file is then checked for a match in the Sybase System 10 SQL server entry. The corresponding port number and hostname are obtained from the *interfaces* file. The */etc/services* file on the *hostname* is checked for a port number match. On the success of this match a connection is established to the Sybase System 10 SQL server.

## 5.4.2 AIX Processes

When you have started the region successfully, issue the following command to see the AIX processes for Sybase System 10:

*ps -ef | grep sybase*

For a local Sybase System 10 implementation you will see:

```
  root  8704 17188   7 17:29:42  pts/9  0:00 grep sybase
sybase 10368 25473   0   Mar 23  pts/6  0:00 -ksh
sybase 18118     1   0   Mar 29  pts/0  0:00 sh ./RUN_SYBASE
sybase 20423 18118   0   Mar 29  pts/0 439:55 /sybase/bin/dataserver
-d/dev/rmasterlv -sSYBASE -e/sybase/install/errorlog -i/s
```

For a remote Sybase System 10 implementation you will see:

```
  root  6928 29948   5 17:32:05  pts/1  0:00 grep sybase
sybase 20156     1   0   Mar 29      -  0:00 sh ./RUN_SYB_SERVER
sybase 24253 20156   2   Mar 29      - 179:34 /sybase/bin/dataserver
-d/dev/rmasterlv -sSYB_SERVER -e/sybase/install/errorlog
sybase 27674 17939   0   Apr 03  pts/3  0:00 -ksh
```

**Note**

Sybase System 10 uses a multithreaded server. Therefore, you do not see any AIX processes corresponding to the CICS/6000 application servers. The AIX *dataserver* process services all requests from CICS/6000. It uses a single thread for each CICS/6000 application server.

# Chapter 6. Problem Determination

In this chapter we briefly describe how to handle problem determination when CICS/6000 is using XA support for DATABASE 2 AIX/6000, Informix Version 5, Oracle7, and Sybase System 10

For more details, refer to the specific database documentation.

## 6.1 DATABASE 2 AIX/6000

When an error is detected during an XA request from the CICS/6000, the user program (the client application and/or server application) might not be able to get the error code from the TM. If your program abends or gets a cryptic return code from the CICS/6000, check the message log file to get more information. The message log file for CICS/6000 is */var/cics_regions/<region_name>/console.msg*. In addition, DATABASE 2 AIX/6000 writes out additional information to the AIX system log if that log is enabled.

To enable the system log for DATABASE 2 AIX/6000 carry out the following steps:

1. Log in as user *root*.
2. Modify the */etc/syslog.conf* file to include the following line:

   *user.warn <file_name>*

   where *<file_name>* is the explicit path of an existing file where the entries will be made (the file must exist before the step 2 is performed). You can create the empty file using the following AIX command:

   *touch <file_name>*
3. Determine the pid of the system log daemon and send a *kill -1* signal to this pid to enable syslogd: *ps -ef | grep syslogd*

   *kill -1 <pid>*

DATABASE 2 AIX/6000 writes all XA-specific errors to the system log as an SQLCA with SQLCODE -998 (transaction or heuristic errors) with the appropriate reason code and subcodes as message tokens. If a connection fails, the connection error or communication error will also have its SQLCA written to the system log.

Some of the common problems we encountered are the following:

- XA open string contains invalid syntax.
- The connection to the database specified in the open string fails because the database has not been cataloged, DATABASE 2 AIX/6000 has not been started, or the server application′s user name or password is not authorized to connect to the database.
- Communication error in client-server configuration
- Cannot read or load the Switch Load file.

Figure 24 is a sample syslog entry for an XA error. The entry is related to an XA open string where instead of using *cicsloc* we made a typing error (*cicslox*).

```
Mar 31 11:44:42 bengal DB2[32056]:
DB2 (db2) XA DTP Support      sqlxa_make_connection reports:
probe id 103 with error 2048 and alert num 0
Mar 31 11:44:42 bengal DB2[32056]:
extra symptom string provided: XA - error connecting to database
Mar 31 11:44:42 bengal DB2[32056]:
data:   53514c43  41202020  00000088  fffffc0b          SQLCA   ....|.
Mar 31 11:44:42 bengal DB2[32056]:
data:   00084349  43534c4f  58ff0000  00000000          ..CICSLOX.......
Mar 31 11:44:42 bengal DB2[32056]:
data:   00000000  00000000  00000000  00000000          ................
Mar 31 11:44:42 bengal last message repeated 2 times
Mar 31 11:44:42 bengal DB2[32056]:
data:   00000000  00000000  53514c45  524c4e4b          ........SQLERLNK
Mar 31 11:44:42 bengal DB2[32056]:
data:   00000000  00000000  00000000  00000000          ................
Mar 31 11:44:42 bengal DB2[32056]:
data:   00000000  00000001  20202020  20202020          ........
Mar 31 11:44:42 bengal DB2[32056]:
data:   20202035  32303035                                 52005
Mar 31 11:44:42 bengal DB2[32056]:
DB2 (db2) XA DTP Support      sqlxa_open reports:
```

*Figure 24. Sample Syslog Entry for XA Error*

You will notice that:

- Each line is prefixed with the time stamp, the product name (DB2), the host name (in this case, bengal) and the pid that reports the problem.
- The line *DB2 (db2) XA DTP Support*

  displays some internal information about where this error was caught. The name in parentheses (db2) is the DB2INSTANCE name.
- The line *extra symptom string provided: XA - error connecting to database*

  informs you that the error occurred in XA connecting a database.
- Text displayed on the right-hand side indicates that this is an SQLCA call so you can find the sqlcode description in the *IBM DATABASE 2 AIX/6000 Messages and Problem Determination Guide*. The sqlcode is hexadecimal fffffc0b (where ffff is negative SQLCODE) and the decimal value is (fc0b-ffff)+1=-1013.

**SQL13N**

The database alias name or database name <*name*> could not be found.

sqlcode=-1013

You can use the Command Line Processor to retreive the message and an explanation of an sqlcode by using the following command:

*db2 ? sql1013*

Figure 25 on page 73 is the */var/cics_regions/<region_name>/console.msg* entry when CICS/6000 is unable to load the Switch Load file, *db2xa*.

```
ERZ10144I/0375 03/31/94 13:42:24 sanjose      : Application server 8 started
ERZ5801E/0005 03/31/94 13:42:30 sanjose      :
Unsuccessful load of program '/var/cics_regions/sanjose/bin/db2xa'; errno 8.
ERZ5802E/0006 03/31/94 13:42:30 sanjose      :
Information on unsuccessful program load: '3 getgrset /usr/lib/libs.a shr.o'.
ERZ1647E/0232 03/31/94 13:42:30 sanjose      :
Abnormal termination A16D:
Unable to load an External Resource Manager XA Support file.
ERZ1040I/0055 03/31/94 13:42:31 sanjose      :
CICS/6000 control process 'cicsas' completed with exit code 0
ERZ5801E/0005 03/31/94 13:42:32 sanjose      :
Unsuccessful load of program '/var/cics_regions/sanjose/bin/db2xa'; errno 8.
ERZ5802E/0006 03/31/94 13:42:32 sanjose      :
Information on unsuccessful program load: '3 getgrset /usr/lib/libs.a shr.o'.
```

*Figure 25. CICS/6000 Console Message When Unable to Load a Switch Load File*

## 6.2 Informix Version 5

Informix Version 5 passes all XA errors to CICS/6000 through the *do_sql_error* routine, and these are entries written into the*/var/cics_regions/<region_name>/console.msg* file.

Some of the common problems we encountered are the following:

- XA open string contains invalid syntax.
- XA open string parameter definition wrong or incomplete.
- Switch Load file not found or available.

Figure 26 is the */var/cics_regions/<region_name>/console.msg* entry when CICS/6000 tries to connect to a nonexistent database using an incorrect open string.

```
ERZ10144I/0375 03/31/94 15:06:50 sanjose      :
Application server 8 started
ERZ8032E/0807 03/31/94 15:07:07 sanjose      :
Abnormal termination U8032. XA_OPEN was unsuccessful when opening
'INFORMIX-ONLINE' because XA_OPEN string
'cicstesx:INFORMIXDIR=/home/informix:TBCONFIG=tbconfig.INF' is invalid.
ERZ1003I/0094 03/31/94 15:07:07 sanjose      :
CICS/6000 is performing region abnormal termination in process 'cicsas'
ERZ5204I/0602 03/31/94 15:07:07 sanjose      : Dump to 'SYSA0001.dmp' started.
```

*Figure 26. CICS/6000 Console Message: Unable to Open Informix Version 5 Database*

We had a problem with the optional field parameter, *TBCONFIG*. This field must be set if your Informix Version 5 OnLine is using a *tbconfig* file different from the default file. In our installation we used the *tbconfig.INF* file. If you do not put that file in the open string, every time you start two CICS/6000 transactions at the same time, the region will dump with the error as shown in Figure 27 on page 74. If you run one transaction at time, however, you will be able to work with the CICS/6000 region.

```
ERZ8030E/8012 03/31/94 16:22:54 sanjose  BA00:
Abnormal termination U8030. XA Interface internal error in ′XA_PRECOM′
by ′INFORMIX-ONLINE′ for XID
′528c0000 0006010c 73616e6a 6f736500 00000006 73616e6a 6f736500 00000006′
(Transaction=′INFU′, TermId=′BA00′, UserId=′gdscics′). ′′. ′
SQLCODE 0, Unknown error message 0.
ERZ1003I/0094 03/31/94 16:22:54 sanjose  BA00: CICS/6000 is performing
region abnormal termination in process ′cicsas′
ERZ5204I/0602 03/31/94 16:22:54 sanjose  BA00: Dump to ′SYSA0001.dmp′ started.
```

*Figure 27. CICS/6000 Console Message: TBCONFIG Not Correctly Set*

It is possible to track the status of global transactions using the Informix Version 5 utility *tbstat* with the *-u* option. This utility is fully documented in the *Informix OnLine Administrator′s Guide*.

## 6.3 Oracle7

The Oracle XA library logs any error and tracing information to its trace file. This information is useful in supplementing the XA error code.

The name of the trace file is:

xa_*db_namedate*.trc,

where *db_name* is the database name you specified in the open string field *DB=db_name*, and *date* is the date when the information was logged to the trace file. If you do not specify *DB=db_name* in the open string, it automatically defaults to the name *NULL*.

Note that multiple Oracle XA library RMs with the same *DB* field and *LogDir* field in their open strings log all trace information that occurs on the same day to the same trace file.

Figure 28 shows an Oracle trace file when a nonexistent user was put in the open string:

```
174221.24447.2:
ORACLE XA: Version 1.0.3.0.0 - Production. RM name = ′Oracle_XA′.

174221.24447.2:
xaolgn: XAER_INVAL; logon denied.
```

*Figure 28. Oracle XA Trace File: Nonexistent User Put in Open String*

The entry in the trace file contains the following information:

- 174221 - time when the information was logged
- 24447 - process ID
- 2 - RM ID
- xaolgn - the module
- logon denied - error information returned.

Figure 29 shows the CICS/6000 console message for the previous error.

```
ERZ8032E/0807 03/31/94 17:42:21 sanjose      :
Abnormal termination U8032. XA_OPEN was unsuccessful when opening ′Oracle_XA′
because XA_OPEN string ′Oracle_XA+Acc=P/scoot/tiger+SesTm=35+LogDir=/tmp′
is invalid.
ERZ8032E/0807 03/31/94 17:42:21 sanjose      :
Abnormal termination U8032. XA_OPEN was unsuccessful when opening ′Oracle_XA′
because XA_OPEN string ′Oracle_XA+Acc=P/scoot/tiger+SesTm=35+LogDir=/tmp′
is invalid.
ERZ1003I/0094 03/31/94 17:42:21 sanjose      :
CICS/6000 is performing region abnormal termination in process ′cicsas′
ERZ5204I/0602 03/31/94 17:42:21 sanjose      : Dump to ′SYSA0001.dmp′ started.
```

*Figure 29. CICS/6000 Console Message: Nonexistent User*

Figure 30 shows the error we received when we tried to configure Oracle7 remotely and we not properly set up the *tnsnames.ora* file.

```
135210.29851.0:
ORACLE XA: Version 1.0.3.0.0 - Production. RM name = ′Oracle_XA′.

135210.29851.0:
xaolgn: XAER_RMERR; upiahs failed. ORA-12203.

135211.29851.0:
ORA-12203: TNS:unable to connect to destination
```

*Figure 30. Sample Oracle XA Trace File: Server Configured Remotely*

Figure 31 on page 76 shows the entry in the Oracle7 XA trace file if you failed to grant the *SELECT* privilege to the *V$XATRANS$* view for all Oracle accounts that Oracle7 XA Library applications will use.

```
175237.24241.1:
ORACLE XA: Version 1.0.3.0.0 - Production. RM name = 'Oracle_XA'.

175237.24241.1:
xaofetch: XAER_RMERR; upipse rtn ORA-942; sql_stmt=SELECT k2gtifmt,
k2gtitid_ext, k2gtibid FROM sys.v$xatrans$

175237.24241.1:
ORA-00942: table or view does not exist

175237.24241.1:
xaorecover: xaofetch rtn -3.
```

*Figure 31. Oracle XA Trace Entry: SELECT Privilege Not Granted to V$XATRANS$ View*

For more details, refer to the *Oracle7 Server for Unix - Administrator's Reference*.

## 6.4 Sybase System 10

Sybase System 10 XA-Library writes tracing information in the fully qualified file name specified in the open string. If you do not specify the -L option and a *logfile* parameter, logging is disabled.

Some of the common problem we encountered are the following:

- XA open string contains invalid syntax.
- The connection to the database specified in the open string fails because the server is not correctly set in the XA configuration file as the LRM.
- Sybase System 10 has not been started.
- User name or password not authorized to connect to the server
- Communication error in client-server configuration
- Cannot read or load Switch Load file.

Figure 32 on page 77 shows a Sybase System 10 trace file if you used the wrong password for user *sa* in the open string.

```
9853:
xa__log_init: pid 9853 started at Fri Apr  1 11:22:35 1994

9853: XA_ENTRY: xa_open: info: -Usa -Psybasi -Nconnection_1
                               -L/tmp/sybase.log rmid: 0x0, flags 0x0
9853: xa__list_open: failed to get XCL name
9853: lrm_lookup: connection_1
9853: lrm_lookup: using config file /sybase/xa_config
9853: lrm_lookup: connection_1
9853: lrm_lookup: using config file /sybase/xa_config
9853: xc__open: Connecting to serv: SYBASE, user: sa, passwd: sybasi.
9853: XA_EXIT: xa_open: ret_stat: -3
27518:
xa__log_init: pid 27518 started at Fri Apr  1 11:22:53 1994
```

*Figure 32. Sybase XA Trace File: Wrong Password for User sa in Open String*

Figure 33 shows the CICS/6000 console message file.

```
ERZ8005E/0805 04/01/94 11:22:37 sanjose      :
Abnormal termination U8005. XA_OPEN returned a Resource Manager error when
opening ′SYBASE_SQL_SERVER′ using XA_OPEN string ′-Usa -Psybasi -Nconnection_1
-L/tmp/sybase.log′. ′′
ERZ1003I/0094 04/01/94 11:22:37 sanjose      :
CICS/6000 is performing region abnormal termination in process ′cicsas′
ERZ5204I/0602 04/01/94 11:22:37 sanjose      : Dump to ′SYSA0001.dmp′ started.
```

*Figure 33. CICS/6000 Console Messages: Wrong Password for User sa in Sybase System 10 Open String*

If you put in your open string an *lrm_name* that is not an entry in the Sybase System 10 *xa_config* file, you will get an entry in the CICS/6000 console message file as shown in Figure 33 and an XA logging trace similar to that shown in Figure 34.

```
25128:
xa__log_init: pid 25128 started at Fri Apr  1 11:45:16 1994

25128: XA_ENTRY: xa_open: info: -Usa -Psybase -Nconnection_3
-L/tmp/log.server rmid: 0x0, flags 0x0
25128: xa__list_open: failed to get XCL name
25128: lrm_lookup: connection_3
25128: lrm_lookup: using config file /sybase/xa_config
25128: lrm_lookup: lrm name not found
25128: xa_list_open: failed to init connection
25128: XA_EXIT: xa_open: ret_stat: -3
31883:
```

*Figure 34. Sybase XA Trace: Wrong LRM Name in Sybase System 10 Open String*

Figure 35 on page 78 shows you the XA trace logging when the server name defined in the *xa_config* entry does not exist.

```
xa__log_init: pid 31883 started at Fri Apr  1 13:49:14 1994

31883: XA_ENTRY: xa_open: info: -Usa -Psybase -Nconnection_2
-L/tmp/log.server rmid: 0x0, flags 0x0
31883: xa__list_open: failed to get XCL name
31883: lrm_lookup: connection_2
31883: lrm_lookup: using config file /sybase/xa_config
31883: lrm_lookup: connection_2
31883: lrm_lookup: using config file /sybase/xa_config
31883: xc__open: Connecting to serv: SYS_SERVER, user: sa, passwd: sybase.
31883: XA_EXIT: xa_open: ret_stat: -3
```

*Figure 35. Sybase XA Trace: Wrong Server Name in Sybase System 10 xa_config File*

# Chapter 7. Application Considerations

In this chapter we describe some of our experiences in configuring CICS/6000 in different scenarios.

Before we actually discuss each scenario, we want you to think about the following points:

- In a DTP environment, server applications are reused by many user transactions, but there is no mechanism for the TM to provide the RM with:
  - The actual user ID that invokes the transaction (all transactions are performed under the server application ID)
  - Information about when a program begins and ends. The RM knows only transactions, and it must treat each transaction as unrelated to the previous transaction.
- The XA Interface allows CICS/6000 to run against different RDBMSs (local or remote) depending on the capabilities of each RDBMS. Some RDBMSs can handle different threads in the same process, but, when used in conjunction with CICS/6000, this feature does not have any advantage. CICS/6000 serializes all operations. For more details you can refer to 2.2.1, “XA Resource Definitions” on page 10. CICS/6000 provides options to serialize the threads so that one thread will execute to a syncpoint before another one begins.
- There is typically a noticeable overhead for crossing between a CICS/6000 transaction program and an SQL database; so the number of crossings should be minimized, for performance reasons. This is a good reason for coding dynamic SQL or calls to SQL stored procedures.
- All updates made through *EXEC SQL* calls in a CICS/6000 transaction can share a common syncpoint with all other updates made elsewhere in the network, under the control of *EXEC CICS SYNCPOINT*.
- CICS/6000 can arrange to drive syncpoint in the database only when the database is actually updated for a given transaction, and only if the database has support for dynamic registration as specified in the X/Open DTP model (2.1.3, “Resource Manager Switch” on page 8). This can reduce the cost of syncpoint when you are running in an environment with multiple XA databases, where transactions usually update only data managed by some of the applications.
- All updates made everywhere by the transaction using *EXEC CICS SYNCPOINT* are committed or rolled back as a unit, no matter how or when networks or components may fail and restart. With a database that supports XA dynamic registration, *EXEC CICS SYNCPOINT* is as efficient as *EXEC SQL COMMIT*, in those cases where it would be safe to use *EXEC SQL COMMIT*.
- Typically, SQL operations that declare cursors or prepare dynamic SQL are relatively expansive, compared with SQL operations that access data; so you should arrange to do the expansive operations no more than once per transaction program.
- Good programming within CICS means keeping your transactions as short lived as possible, utilizing such CICS methods as pseudo conversational programming to present the illusion of a long-running task.

- In CICS/6000 a single application server (*cicsas*) process is bound to a particular task for the duration of the task. If the task suspends itself or performs I/O or a similar operation that may allow it to be context switched, it remains loaded in the application server. The application server was designed to minimize process creation and initialization time.

## 7.1 CICS/6000 and a Local RDBMS

This scenario is probably the easiest but the worst from the point of view of resource distribution. The database machine would be a dedicated machine in order to obtain distribution of tasks on specialized hardware.

Each single RDBMS would be allowed to define a different database on a single instance; below we analyze the capabilities of each RDBMS to handle this feature using the XA Interface.

### 7.1.1 CICS/6000 Transaction and DATABASE 2 AIX/6000

DATABASE 2 AIX/6000 allows you to create several databases running on one instance of the database manager. Creating a database sets up all system catalog tables that the database needs and allocates the database recovery log.

For each database used, CICS/6000 requires an entry into the XA Definition stanza as shown in Figure 8 on page 22, using the name of the database as the connection string. You can imagine writing an application that can switch from one database to another using explicitly the SQL *CONNECT* statement as in the fragment of the uxa1.sqc sample program shown in Figure 36 on page 81.

```
#define DATABASE1 ″cicsloc″
#define DATABASE2 ″cicsrem″

EXEC SQL BEGIN DECLARE SECTION;

char dbase[15];

EXEC SQL END DECLARE SECTION;

main()
{
        strcpy(dbase,DATABASE1);
        EXEC SQL CONNECT TO :dbase;
                       .
                       .
                       .
            EXEC SQL UPDATE cheese
            set order_quantity = :order_quantity
            where name = :name;
                       .
                       .
                       .

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

        strcpy(dbase,DATABASE2);
        EXEC SQL CONNECT TO :dbase;
                       .
                       .
                       .
            EXEC SQL UPDATE cheese
            set order_quantity = :order_quantity
            where name = :name;
                       .
                       .
                       .
}
```

*Figure 36. Connect in SQL C Program*

**Note**

DATABASE 2 AIX/6000 requires an explicit disconnect before you can connect to a different database. In a CICS/6000 application using the XA library you never do an *EXEC SQL DISCONNECT* (it will force immediate shutdown of the CICS/6000 region), but you must execute an *EXEC CICS SYNCPOINT* otherwise the new connection does not take effect.

If you do not explicitly connect to a database in your application, CICS/6000 assumes the last succesful connection. If you try to run against a database an application that does not belong to it, you will get a time-stamp conflict error:

*SQL0818N A timestamp conflict occurred. SQLSTATE=51003 sqlcode=-818.*

Your application must be precompiled and bound on both databases (remote and local).

Upon any successful *SQL CONNECT* (including a *CONNECT* with no options, which only queries the current connection) a number will be returned in the SQLERR[5] field in the SQLCA structure. The possible values are:

- -**0**-No information (value returned only by DB2/2 1.0 and earlier releases - included here for reference).
- -**1**- The connection does not allow syncpoint flow but allows updates (non-XA transaction).
- -**2**- The connection does not allow syncpoint flow or updates (global transaction connecting to a DRDA database).
- -**3**- The connection allows syncpoint flow (global transaction connecting to DATABASE 2 AIX/6000).
- -**4**- Reserved.

The database manager allows the use of the *SQL CONNECT* with the *USER/USING* clause to explicitly establish an SQL access ID different from the access ID of the server application.

The following restrictions apply:

- *SQL COMMIT* and *ROLLBACK* statements not allowed

  The error could be SQLSTATE 56021, 56028, 56029, depending on whether it was a static or dynamic commit or rollback statement.

- Handling database-initiated rollback

  If the RM initiated a rollback, all subsequent SQL requests will be rejected with SQLSTATE 51021 to inform you that you must roll back the global transaction with the TM′s syncpoint services - CICS/6000 *CICS SYNCPOINT ROLLBACK*.

- No cursors declared with the WITH HOLD option

  DATABASE 2 AIX/6000 rejects any attempt to open a WITH HOLD cursor in the CICS/6000 environment with SQLSTATE 56026.

- No updates allowed to a DRDA database
- API restrictions

  APIs are not supported in a DTP environment because they would internally issue a commit in the database and bypass the two-phase commit process (SQLSTATE 56026):

  *Precompile Bind Backup Restore Restart Roll Forward Migrate Import Export Reorganize Table*

For more details see the *IBM DATABASE 2 AIX/6000 Programming Guide* and the *AIX CICS/6000 Application Programming Guide.*

In Figure 37 on page 83 you can see how we modified the original makefile for the *UXA1* transaction in order to update the same row in the same table into two different DATABASE 2 AIX/6000 databases.

```
all: uxa1m.map db2_lr

uxa1m.map: uxa1.bms
    cicsmap uxa1.bms

db2_lr: db2_lr.ccs
    CCFLAGS="/usr/lib/db2.o"; \
    export CCFLAGS; \
    cicstcl -e -d -lC db2_lr.ccs

db2_lr.ccs: db2_lr.sqc
    db2 connect to cicsloc; \
    db2 prep db2_lr.sqc bindfile package using db2_lr; \
    db2 bind db2_lr.bnd ;\
    db2 grant execute on package db2_lr to public
    db2 connect to cicsrem; \
    db2 bind db2_lr.bnd ;\
    db2 grant execute on package db2_lr to public
    mv db2_lr.c db2_lr.ccs
```

*Figure 37. Sample Makefile to Compile a Program Using Two Different DATABASE 2 AIX/6000 Databases*

## 7.1.2 CICS/6000 Transaction and Local Informix Version 5

The only configuration actually supported by Informix Version 5 is the local configuration. It is not possible to configure more than one database running on the same instance of Informix Version 5 in CICS/6000 XAD stanza. If you try to configure more than one, the CICS/6000 region is terminated during cold start, and Informix Version 5 reports an internal error inside the */var/cics_regions/<region name>/console.msg* file. After this error you have to stop Informix Version 5 and start again in order to bring up the CICS/6000 region.

The following SQL statements behave differently in an X/Open DTP environment:

- *CLOSE DATABASE*
- *CREATE DATABASE*
- *DATABASE*
- *LOCK TABLE*
- *SET ISOLATION*
- *SET EXPLAIN*
- *SET LOCK MODE*
- *SET LOG*
- *UNLOCK TABLE*
- *BEGIN WORK*
- *COMMIT WORK*
- *ROLLBACK WORK*.

For more details refer to the *Informix TP/XA User Manual*.

We tried using the *DATABASE* statement inside the code to switch to another database and we got the following SQL error message:

*-701 Statement is invalid within the XA environment*

as documented in the *Informix Error Messages* manual.

### 7.1.3 CICS/6000 Transaction and Oracle7

Oracle7 could support both local and remote databases and it is also possible to write applications in a CICS/6000 environment that can access different Oracle7 databases. An application may include the default database, as well as one or more named databases respecting the following rules:

- Each database must have an open string configured to CICS/6000.
- The default database open string must be without the *DB* optional field.
- The other database open string must specify *DB=db_name*.
- The application server program must contain the SQL DECLARE DATABASE statement for each nondefault database.

Figure 38 shows how you can access three Oracle7 databases in one application server (one default, Oracle1, and Oracle2).

```
EXEC SQL DECLARE Oracle1 DATABASE;
EXEC SQL DECLARE Oracle2 DATABASE;


EXEC SQL BEGIN DECLARE SECTION;
    db_name1 CHARACTER(10);
    db_name2 CHARACTER(10);
EXEC SQL END DECLARE SECTION;
          .
          .
          .
set db_name1 = 'Oracle1';
set db_name2 = 'Oracle2';
          .
          .
          .
 EXEC SQL UPDATE cheese
 set order_quantity = :order_quantity where name = :name;

 EXEC SQL AT :db_name1 UPDATE cheese
 set order_quantity = :order_quantity where name = :name;

 EXEC SQL AT :db_name1 UPDATE cheese
 set order_quantity = :order_quantity where name = :name;
```

*Figure 38. Using Different Oracle7 Databases in One Application Server*

**Note**

The names specified in each open string *DB* optional field must be the same as those used in the SQL DECLARE DATABASE statement (in this example, Oracle1 and Oracle2).

Applications using XA do not create Oracle database connections of their own. Any work that they perform would be outside the global transaction and may confuse the connection information used by the Oracle7 XA library.

Oracle XA applications can access other Oracle7 databases through a database link, with the following restrictions:

- The other database accessed should be another Oracle7.
- Access to the other database must use SQL*NET V2.
- The following SQL statements must never be used:
  - *EXEC SQL ROLLBACK WORK*
  - *EXEC SQL COMMIT WORK*
  - DDL SQL statement like *CREATE TABLE*
  - *EXEC SQL SAVEPOINT*
  - *SET TRANSACTION READ ONLY*
  - *EXEC SQL WORK RELEASE*
  - *EXEC SQL ROLLBACK WORK RELEASE*.

Cursors, when used in Oracle XA applications, are valid only for the duration of the transaction. Explicit cursors should be opened after the transaction begins and closed before the commit or rollback. Also, you must use the *release_cursor=yes* option when compiling your precompiler application.

You can use precompilers with the default database or with named databases as shown in Figure 38 on page 84. Only one default connection is allowed per process.

**Note**

The Oracle7 *libsharesqlxa.a* must be linked to your application. Verify that you are using the *libsharesqlxa.a* that you copied in the */usr/lib* directory as documented in 3.4.2.1, “Shared Library” on page 32.

### 7.1.4 CICS/6000 Transaction and Sybase System 10

Sybase System 10 supports both local and remote databases. Embedded SQL applications running in a CICS/6000 region must conform to the following constraints in order to function within the Sybase System 10 XA environment:

- Transaction management

  The CICS/6000 region is responsible for transaction management, so applications operating in this environment cannot issue SQL statements that manage *begin, commit, and rollback* transactions.

- Connection management

  In the CICS/6000 region, applications rely on the Sybase System 10 XA environment for management of client-server connections. Connection management occurs transparently to the application, so it cannot invoke *CONNECT and DISCONNECT*.

- Current connection

  The notion of default connection does not exist in the Sybase System 10 XA environment, so applications must always explicitly specify a current connection. There are two ways to specify the current connection: through the *SET CONNECTION* command or the *AT CONNECTION name* clause.

  A current connection does not span transactions; an application must reset the current connection after each *EXEC CICS SYNCPOINT* command.

Figure 39 shows how you can use the *set connection* command.

```
                     .
                     .
    EXEC SQL WHENEVER NOT FOUND GOTO :errexit;

    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;
                     .
                     .
```

*Figure 39. Using the SET CONNECTION Command*

You can find more information in the *Sybase System 10 XA-Library Integration Guide for CICS/6000*.

## 7.1.5 CICS/6000 Transaction and Multiple Databases

We modified the *UXA1* transaction to update the same row in the same table of DATABASE 2 AIX/6000, Informix Version 5, Oracle7, and Sybase System 10. The three different updates belong to the same logical unit of work (LUW).

**Note**

We tried the program shown in Figure 40 on page 88 with Informix OnLine Version 5.01, but we always got an SQL return code indicating an error not documented (unrecognized *sqlca.sqlcode*).

Moving to Informix Online Version 5.02 we managed to solve the problem partially. The program worked fine as long as it accessed an Informix database with the *SQL SELECT* statement first; otherwise we still got an SQL unknown return code. The problem did not occur with *SQL UPDATE* statement.

The implementation steps discussed in 3.3.2, “Informix Version 5 Configuration” on page 24 were tested with Informix OnLine Version 5.01, although there are no differences for this implementation between Version 5.01 and Version 5.02.

Each database uses its own precompiler, so the SQL statements are in different programs. Figure 40 on page 88 shows the main file (*alldb1.ccs*) and the select and/or update routines for each database. This sample is not the best programming style, but our aim was to:

- Determine whether we could use multiple databases in a single CICS/6000 transaction.
- Give an example of a method to compile and link a program that requires multiple database preprocessors.

File **alldb1.ccs**

```
#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>

#define UPDATECHEESE "The cheese tables were successfully updated"

int rcode;
int retcode;

char name[16];
char supplier[30];
char supplier_address[30];
int order_quantity;


extern syb_select();
extern syb_update();
extern db2_select();
extern db2_update();
extern inf_select();
extern inf_update();
extern ora_select();
extern ora_update();

main()
{
        char errmsg[400];
        char qmsg[400];
    short mlen;

        /*   Get addresability for EIB */

        EXEC CICS ADDRESS EIB(dfheiptr);

    /*   Write record to CICS temporary storage queue  */

    sprintf(qmsg, "%s", "Running Transaction MXA1");
    mlen = strlen(qmsg);
    EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                     LENGTH(mlen) RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X000");

        /* Send the first map */

        EXEC CICS SEND MAP("PANEL1") MAPSET("MXA1")
                       FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X001");
```

*Figure 40 (Part 1 of 21). Sample Transaction Involving Different Databases*

```
        /* Receive the response */

        EXEC CICS RECEIVE MAP("PANEL1") MAPSET("MXA1") 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);


    fprintf(stderr,"%d: cheese name is %s\n",getpid(),name);

            retcode = inf_select(name);
            if (retcode == 0)
            {
          sprintf(qmsg, "%s,%s", "INFORMIX: cheese present: ",name);
          mlen = strlen(qmsg);
          EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                      LENGTH(mlen) RESP(rcode);
          if (rcode != DFHRESP(NORMAL))
                     EXEC CICS ABEND ABCODE("X007");

            }

            retcode = syb_select(name);
            if (retcode == 0)
            {
          sprintf(qmsg, "%s,%s", "SYBASE: cheese present: ",name);
          mlen = strlen(qmsg);
          EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                      LENGTH(mlen) RESP(rcode);
          if (rcode != DFHRESP(NORMAL))
                     EXEC CICS ABEND ABCODE("X007");

            }

            retcode = db2_select(name);
            if (retcode == 0)
            {
          sprintf(qmsg, "%s,%s", "DB2/6000: cheese present: ",name);
          mlen = strlen(qmsg);
          EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                      LENGTH(mlen) RESP(rcode);
          if (rcode != DFHRESP(NORMAL))
                     EXEC CICS ABEND ABCODE("X007");

            }

            retcode = ora_select(name);
            if (retcode == 0)
            {
          sprintf(qmsg, "%s,%s", "ORACLE: cheese present: ",name);
          mlen = strlen(qmsg);
          EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                      LENGTH(mlen) RESP(rcode);
```

*Figure 40 (Part 2 of 21). Sample Transaction Involving Different Databases*

```
       if (rcode != DFHRESP(NORMAL))
                  EXEC CICS ABEND ABCODE("X007");

         }

     /* Fill in and send the second map */

     EXEC CICS SEND MAP("PANEL2") MAPSET("MXA1")
                    FREEKB ERASE RESP(rcode);
     if (rcode != DFHRESP(NORMAL))
                    EXEC CICS ABEND ABCODE("X003");

     /* Receive the response */

     EXEC CICS RECEIVE MAP("PANEL2") MAPSET("MXA1") RESP(rcode);
     if (rcode != DFHRESP(NORMAL))
         EXEC CICS ABEND ABCODE("X004");


  fprintf(stderr,"reply was %x\n", panel2.panel2i.questi);
 if (panel2.panel2i.questi == 'y')
 {

         /* Send the third map   */
         EXEC CICS SEND MAP("PANEL3") MAPSET("MXA1")
                        FREEKB ERASE RESP(rcode);

         if (rcode != DFHRESP(NORMAL))
             EXEC CICS ABEND ABCODE("X005");
         /* Receive the response */

         EXEC CICS RECEIVE MAP("PANEL3") MAPSET("MXA1") RESP(rcode);
         if (rcode != DFHRESP(NORMAL))
             EXEC CICS ABEND ABCODE("X006");

         /* Update the Database */

         order_quantity = atoi(panel3.panel3i.newordi);

         retcode = syb_update(name,order_quantity);
         if (retcode == 0)
         {
       sprintf(qmsg, "%s", "SYBASE: cheese table updated ");
       mlen = strlen(qmsg);
       EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                    LENGTH(mlen) RESP(rcode);
       if (rcode != DFHRESP(NORMAL))
                  EXEC CICS ABEND ABCODE("X007");

         }
```

*Figure 40 (Part 3 of 21). Sample Transaction Involving Different Databases*

```
          retcode = db2_update(name,order_quantity);
          if (retcode == 0)
          {
        sprintf(qmsg, "%s", "DB2/6000: cheese table updated");
        mlen = strlen(qmsg);
        EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                  LENGTH(mlen) RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                 EXEC CICS ABEND ABCODE("X007");

          }

          retcode = inf_update(name,order_quantity);
          if (retcode == 0)
          {
        sprintf(qmsg, "%s", "INFORMIX: cheese table updated");
        mlen = strlen(qmsg);
        EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                  LENGTH(mlen) RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                 EXEC CICS ABEND ABCODE("X007");

          }

          retcode = ora_update(name,order_quantity);
          if (retcode == 0)
          {
        sprintf(qmsg, "%s", "ORACLE: cheese table updated");
        mlen = strlen(qmsg);


        EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                  LENGTH(mlen) RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                 EXEC CICS ABEND ABCODE("X007");

          }

          /* Write a record to the temporary queue */

      sprintf(qmsg, "%s", "The cheese tables were updated");
      mlen = strlen(qmsg);
        EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg)
                  LENGTH(mlen) RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                 EXEC CICS ABEND ABCODE("X007");

          }
  else
  {

          /* The user does not wish to update so free the keyb'd & return  */
      fprintf(stderr,"not a y, a %c\n", panel2.panel2i.questi);
```

*Figure 40 (Part 4 of 21). Sample Transaction Involving Different Databases*

```
            EXEC CICS SEND CONTROL ERASE FREEKB;
            EXEC CICS 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  */

        sprintf(panel4.panel4o.messageo, UPDATECHEESE);
        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                       FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
            EXEC CICS ABEND ABCODE("X007");

            /* free the keyb'd & return  */

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

*Figure 40 (Part 5 of 21). Sample Transaction Involving Different Databases*

```
File inf_sel.ec


#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>


#define NOCHEESE "There is no such cheese in the table"
#define SQLNOTFOUND 100

extern int inf_select();
EXEC SQL INCLUDE sqlca;

int inf_select(db_cheese)
char db_cheese[15];
{
        char errmsg[400];
        char qmsg[400];
    int rcode;
    int mlen;

    EXEC SQL BEGIN DECLARE SECTION;

    int order_quantity;
    varchar name[15];
    char supplier[30];
    char supplier_address[30];

    EXEC SQL END DECLARE SECTION;

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

    strcpy(name,db_cheese);

    EXEC SQL SELECT order_quantity
    into
        :order_quantity
    FROM cheese
    WHERE name = :name;

        sprintf(panel2.panel2o.order3o, "%d", order_quantity);
    return(0);

errexit:

    EXEC SQL WHENEVER SQLERROR CONTINUE;

        /* Handle "no rows returned" from SELECT */
         if (sqlca.sqlcode == SQLNOTFOUND)
         {
              sprintf(panel4.panel4o.messageo, "%s", NOCHEESE);
```

*Figure 40 (Part 6 of 21). Sample Transaction Involving Different Databases*

```
                    EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                            FREEKB ERASE RESP(rcode);
                    if (rcode != DFHRESP(NORMAL))
                         EXEC CICS ABEND ABCODE("X009");

                    EXEC CICS SEND CONTROL FREEKB;
                    EXEC CICS RETURN;
          }

     rgetmsg((short)sqlca.sqlcode, errmsg, sizeof(errmsg));
     strncpy(panel4.panel4o.messageo, errmsg, 60);
         sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);

         /* Send the fourth map with appropriate message */

         EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                      FREEKB ERASE RESP(rcode);
         if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

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

*Figure 40 (Part 7 of 21). Sample Transaction Involving Different Databases*

```
File syb_sel.cpre


#include <stdio.h>
#include <sybtesql.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>

#define SQLNOTFOUND 100
#define NOCHEESE "There is no such cheese in the table"

extern int syb_select();

EXEC SQL INCLUDE sqlca;

int syb_select(db_cheese)
char db_cheese[16];
{


int rcode;
char errmsg[400];


EXEC SQL BEGIN DECLARE SECTION;

char name[16];
char supplier[30];
char supplier_address[30];
int order_quantity;

EXEC SQL END DECLARE SECTION;
    EXEC SQL WHENEVER SQLERROR GOTO :errexit;
    EXEC SQL WHENEVER NOT FOUND GOTO :errexit;
    EXEC SQL SET CONNECTION connection_1;

    strcpy(name,db_cheese);

    EXEC SQL SELECT name, supplier, supplier_address, order_quantity
    into
        :name, :supplier, :supplier_address, :order_quantity
    FROM cheese
    WHERE name = :name;


        sprintf(panel2.panel2o.nameo, "%s", name);
        sprintf(panel2.panel2o.supplo, "%s",supplier);
        sprintf(panel2.panel2o.addresso, "%s", supplier_address);
        sprintf(panel2.panel2o.order1o, "%d", order_quantity);

    return(0);
```

*Figure 40 (Part 8 of 21). Sample Transaction Involving Different Databases*

```
errexit:
        /* Handle "no rows returned" from SELECT */
          if (sqlca.sqlcode == SQLNOTFOUND)
          {
                sprintf(panel4.panel4o.messageo, "%s", NOCHEESE);
                      EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                         FREEKB ERASE RESP(rcode);
                      if (rcode != DFHRESP(NORMAL))
                           EXEC CICS ABEND ABCODE("X009");
                      EXEC CICS SEND CONTROL FREEKB;
                      EXEC CICS RETURN;
          }

        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 */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                       FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

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

*Figure 40 (Part 9 of 21). Sample Transaction Involving Different Databases*

File **db2_sel.sqc**

```
#include <sql.h>
#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>

#define NOCHEESE "There is no such cheese in the table"
#define SQLNOTFOUND 100

extern int db2_select();
EXEC SQL INCLUDE sqlca;

int db2_select(db_cheese)
char db_cheese[15];
{
        char errmsg[400];
    int rcode;

    EXEC SQL BEGIN DECLARE SECTION;

    long int order_quantity;
    char name[16];
    char supplier[30];
    char supplier_address[30];

    EXEC SQL END DECLARE SECTION;

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

    strcpy(name,db_cheese);

    EXEC SQL SELECT order_quantity
    into
        :order_quantity
    FROM cheese
    WHERE name = :name;

        sprintf(panel2.panel2o.order2o, "%d", order_quantity);
    return(0);

errexit:

        /* Handle "no rows returned" from SELECT */
         if (sqlca.sqlcode == SQLNOTFOUND)
         {
                sprintf(panel4.panel4o.messageo, "%s", NOCHEESE);
                     EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                              FREEKB ERASE RESP(rcode);
                     if (rcode != DFHRESP(NORMAL))
                           EXEC CICS ABEND ABCODE("X009");
```

*Figure 40 (Part 10 of 21). Sample Transaction Involving Different Databases*

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

        sqlaintp (errmsg, sizeof(errmsg), 0, &sqlca);
    strncpy(panel4.panel4o.messageo, errmsg, 60);
        sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);

        /* Send the fourth map with appropriate message */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1") FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

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

*Figure 40 (Part 11 of 21). Sample Transaction Involving Different Databases*

File **ora_sel.pc**

```
#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>

#define NOCHEESE "There is no such cheese in the table"
#define SQLNOTFOUND 1403

extern int ora_select();


EXEC SQL INCLUDE sqlca;


int ora_select(db_cheese)
char db_cheese[16];
{
        char errmsg[400];
    int rcode;

    EXEC SQL BEGIN DECLARE SECTION;

    varchar name[16];
    int order_quantity;
    char supplier[30];
    char supplier_address[30];

    EXEC SQL END DECLARE SECTION;

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

    strcpy(name.arr,db_cheese);
    name.len = strlen(name.arr);

    EXEC SQL SELECT order_quantity
    into
        :order_quantity
    FROM cheese
    WHERE NAME = :name;

        sprintf(panel2.panel2o.order4o, "%d", order_quantity);

    return(0);

errexit:
        /* Handle "no rows returned" from SELECT */
         if (sqlca.sqlcode == SQLNOTFOUND)
         {
               sprintf(panel4.panel4o.messageo, "%s", NOCHEESE);
                     EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1")
                               FREEKB ERASE RESP(rcode);
                     if (rcode != DFHRESP(NORMAL))
                           EXEC CICS ABEND ABCODE("X009");

                     EXEC CICS SEND CONTROL FREEKB;
```

*Figure 40 (Part 12 of 21). Sample Transaction Involving Different Databases*

```
                    EXEC CICS RETURN;
            }

        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 */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1") FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

        /* Rollback the transaction */

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

*Figure 40 (Part 13 of 21). Sample Transaction Involving Different Databases*

File **syb_up.cpre**

```
#include <stdio.h>
#include <sybtesql.h>
#include <cics_packon.h>
#include ″mxa1.h″
#include <cics_packoff.h>

#define NOCHEESE ″There is no such cheese in the table″

extern int syb_update();

EXEC SQL INCLUDE sqlca;

int syb_update(db_name,db_order_quantity)
char db_name[16];
int db_order_quantity;
{
        char errmsg[400];
        char qmsg[400];
    short mlen;
    int rcode;
    int retcode;

    EXEC SQL BEGIN DECLARE SECTION;

    char name[16];
    int order_quantity;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL WHENEVER SQLERROR GOTO :errexit;

    strcpy(name,db_name);
    order_quantity = db_order_quantity;


            /* Update the Database */

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

    sprintf(qmsg, ″%s″, ″UPDATE SYBASE″);
    mlen = strlen(qmsg);
    EXEC CICS WRITEQ TS QUEUE(″TEMPXAQ1″) FROM(qmsg) LENGTH(mlen) RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
          EXEC CICS ABEND ABCODE(″X007″);

        return(0);

errexit:

        sprintf(errmsg, ″%.60s\n″, sqlca.sqlerrm.sqlerrmc);
```

*Figure 40 (Part 14 of 21). Sample Transaction Involving Different Databases*

```
    strncpy(panel4.panel4o.messageo, errmsg, 60);
        sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);

        /* Send the fourth map with appropriate message */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1") FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

        /* Rollback the transaction */

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

*Figure 40 (Part 15 of 21). Sample Transaction Involving Different Databases*

File **db2_up.sqc**

```
#include <sql.h>
#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>

#define NOCHEESE "There is no such cheese in the table"

extern int db2_update();
EXEC SQL INCLUDE sqlca;

int db2_update(db_name,db_order_quantity)
char db_name[15];
int db_order_quantity;
{
        char errmsg[400];
        char qmsg[400];
    short mlen;
    int rcode;
    int retcode;

    EXEC SQL BEGIN DECLARE SECTION;

    char name[15];
    long int order_quantity;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL WHENEVER SQLERROR GOTO :errexit;


    strcpy(name,db_name);
    order_quantity = db_order_quantity;


            /* Update the Database */

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


    sprintf(qmsg, "%s", "UPDATE DB2/6000 ");
    mlen = strlen(qmsg);
    EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg) LENGTH(mlen) RESP(rcode);
         if (rcode != DFHRESP(NORMAL))
                   EXEC CICS ABEND ABCODE("X007");

        return(0);

errexit:
```

*Figure 40 (Part 16 of 21). Sample Transaction Involving Different Databases*

```
        sqlaintp(errmsg, sizeof(errmsg), 0, &sqlca);

    strncpy(panel4.panel4o.messageo, errmsg, 60);
        sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);


        /* Send the fourth map with appropriate message */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1") FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

        /* Rollback the transaction */

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

*Figure 40 (Part 17 of 21). Sample Transaction Involving Different Databases*

File **inf_up.ec**

```
#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>

#define NOCHEESE "There is no such cheese in the table"

extern int inf_update();

EXEC SQL INCLUDE sqlca;

int inf_update(db_name,db_order_quantity)
char db_name[15];
int db_order_quantity;
{
        char errmsg[400];
        char qmsg[400];
    short mlen;
    int rcode;
    int retcode;

    EXEC SQL BEGIN DECLARE SECTION;

    varchar name[15];
    int order_quantity;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL WHENEVER SQLERROR GOTO :errexit;

    strcpy(name,db_name);
    order_quantity = db_order_quantity;


            /* Update the Database */

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


    sprintf(qmsg, "%s", "UPDATE INFORMIX");
    mlen = strlen(qmsg);
    EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg) LENGTH(mlen) RESP(rcode);
          if (rcode != DFHRESP(NORMAL))
                    EXEC CICS ABEND ABCODE("X007");

        return(0);
```

*Figure 40 (Part 18 of 21). Sample Transaction Involving Different Databases*

```
errexit:

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    rgetmsg(sqlca.sqlcode, errmsg, sizeof(errmsg));

    strncpy(panel4.panel4o.messageo, errmsg, 60);
        sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);

        /* Send the fourth map with appropriate message */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1") FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

        /* Rollback the transaction */

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

*Figure 40 (Part 19 of 21). Sample Transaction Involving Different Databases*

File **ora_up.pc**

```
#include <stdio.h>
#include <cics_packon.h>
#include "mxa1.h"
#include <cics_packoff.h>


#define NOCHEESE "There is no such cheese in the table"

extern int ora_update();


EXEC SQL INCLUDE sqlca;


int ora_update(db_name,db_order_quantity)
char db_name[16];
int db_order_quantity;
{
        char errmsg[400];
        char qmsg[400];
    short mlen;
    int rcode;
    int retcode;


    EXEC SQL BEGIN DECLARE SECTION;

    varchar name[16];
    int order_quantity;

    EXEC SQL END DECLARE SECTION;

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

    strcpy(name.arr,db_name);
    name.len = strlen(name.arr);

    order_quantity = db_order_quantity;

            /* Update the Database */

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


    sprintf(qmsg, "%s", "UPDATE ORACLE");
    mlen = strlen(qmsg);
    EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1") FROM(qmsg) LENGTH(mlen) RESP(rcode);
```

*Figure 40 (Part 20 of 21). Sample Transaction Involving Different Databases*

```
        if (rcode != DFHRESP(NORMAL))
          EXEC CICS ABEND ABCODE("X007");
        return(0);

errexit:
        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 */

        EXEC CICS SEND MAP("PANEL4") MAPSET("MXA1") FREEKB ERASE RESP(rcode);
        if (rcode != DFHRESP(NORMAL))
                EXEC CICS ABEND ABCODE("X008");

        /* Rollback the transaction */

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

*Figure 40 (Part 21 of 21). Sample Transaction Involving Different Databases*

Figure 41 on page 109 shows the makefile for compiling and linking our sample program.

```
#
# Makefile to build the MXA1 example program using Sybase Oracle
# Informix e DB2/6000
#

SHAROBJS = -lcs.so -lcomn.so

SYB_SHAR_LIBDIR = $(SYBASE)/lib

LIBDIR = $(SYBASE)/lib

SYBLIBS = -lct -lintl -ltcl -lm -linsck
CFLAGS = -c -g


all: mxa1m.map all1

mxa1m.map: mxa1.bms
    cicsmap mxa1.bms

all1:syb_sel.o syb_up.o db2_sel.o db2_up.o inf_sel.o inf_up.o ora_sel.o
     ora_up.o alldb.c
    xlc_r -v -emain alldb&dotc -o all1 syb_sel.o syb_up.o db2_sel.o \
    db2_up.o inf_sel.o inf_up.o ora_sel.o ora_up.o \
    -I/usr/lpp/cics/v1.1/include \
    -L/usr/lib \
    -L $(SYB_SHAR_LIBDIR)\
    -L/usr/lpp/cics/v1.1/lib  \
    -L $(LIBDIR) \
    -L/home/oracle/lib \
    $(SHAROBJS) \
    $(SYBLIBS) \
    /usr/lpp/cics/v1.1/lib/libcics_api.a \
    /usr/lib/db2.o \
    /usr/lib/libinf501cs.o \
    -lsharesqlxa -lm -lld

inf_sel.o:inf_sel.ccs
    cicstran -e -d -lC inf_sel.ccs
    xlc_r -v -c inf_sel.c \
    -I/usr/lpp/cics/v1.1/include -I$(INFORMIXDIR)/incl/esql

inf_up.o:inf_up.ccs
    cicstran -e -d -lC inf_up.ccs
    xlc_r -v -c inf_up.c \
    -I/usr/lpp/cics/v1.1/include -I$(INFORMIXDIR)/incl/esql

syb_sel.o:syb_sel.ccs
    cicstran -e -d -lC syb_sel.ccs
    xlc_r -v -c syb_sel.c \
    -I/usr/lpp/cics/v1.1/include -I$(SYBASE)/include
```

*Figure 41 (Part 1 of 3). Sample Makefile for Compiling a Program That Uses Different Databases*

```
syb_up.o:syb_up.ccs
    cicstran -e -d -lC syb_up.ccs
    xlc_r -v -c syb_up.c \
    -I/usr/lpp/cics/v1.1/include -I$(SYBASE)/include

ora_sel.o:ora_sel.ccs
    cicstran -e -d -lC ora_sel.ccs
    xlc_r -v -c ora_sel.c \
    -I/usr/lpp/cics/v1.1/include -I$(ORACLE_HOME)/include

ora_up.o:ora_up.ccs
    cicstran -e -d -lC ora_up.ccs
    xlc_r -v -c ora_up.c \
    -I/usr/lpp/cics/v1.1/include -I$(ORACLE_HOME)/include

db2_sel.o: db2_sel.ccs
    cicstran -e -d -lC db2_sel.ccs
    xlc_r -v -c db2_sel.c \
    -I/usr/lpp/cics/v1.1/include

db2_up.o: db2_up.ccs
    cicstran -e -d -lC db2_up.ccs
    xlc_r -v -c db2_up.c \
    -I/usr/lpp/cics/v1.1/include

db2_sel.ccs:db2_sel.sqc
    db2 connect to cicsloc; \
    db2 prep db2_sel.sqc; \
    db2 grant execute on package db2_sel to public
    mv db2_sel.c db2_sel.ccs

db2_up.ccs:db2_up.sqc
    db2 connect to cicsloc; \
    db2 prep db2_up.sqc; \
    db2 grant execute on package db2_up to public
    mv db2_up.c db2_up.ccs

inf_sel.ccs:inf_sel.ec
    $(INFORMIXDIR)/bin/esql -e inf_sel.ec
    mv inf_sel.c inf_sel.ccs

inf_up.ccs:inf_up.ec
    $(INFORMIXDIR)/bin/esql -e inf_up.ec
    mv inf_up.c inf_up.ccs

syb_sel.ccs:syb_sel.cpre
    $(SYBASE)/bin/cpre syb_sel.cpre
    mv syb_sel.c syb_sel.ccs

syb_up.ccs:syb_up.cpre
    $(SYBASE)/bin/cpre syb_up.cpre
    mv syb_up.c syb_up.ccs
```

*Figure 41 (Part 2 of 3). Sample Makefile for Compiling a Program That Uses Different Databases*

```
ora_sel.ccs:ora_sel.pc
    $(ORACLE_HOME)/bin/proc release_cursor=yes \
sqlcheck=none ireclen=512 iname=ora_sel.pc
    mv ora_sel.c ora_sel.ccs

ora_up.ccs:ora_up.pc
    $(ORACLE_HOME)/bin/proc release_cursor=yes \
sqlcheck=none ireclen=512 iname=ora_up.pc
    mv ora_up.c ora_up.ccs

alldb.c:alldb.ccs
    cicstran -e -d -lC alldb.ccs
```

*Figure 41 (Part 3 of 3). Sample Makefile for Compiling a Program That Uses Different Databases*

# Appendix A. Appendix A. MicroFocus COBOL Run-Time Support

In this appendix we briefly describe how to create a MicroFocus COBOL run time for CICS/6000 and RDBMSs.

To run a COBOL program you must first create in */usr/lpp/cics/v1.1/bin* a file called *cicsprCOBOL*, which contains the COBOL support routines and run-time information.

To create the COBOL run-time support for CICS/6000 you have to carry out the following steps:

1. Log in as user root.
2. Install Micro Focus COBOL for UNIX (Release 3.1 or later).
3. Set the *COBDIR* environment variable to the directory where MicroFocus COBOL is installed (on our machine, */usr/lib/cobol*)
4. Add *$COBDIR/bin* and */usr/lpp/cics/v1.1/bin* to the *PATH* environment variable.
5. Change to the proper directory by entering:

   *cd /usr/lpp/cics/v1.1/bin*.
6. Run the command *cicsmkcobol* [**-o** *OutPutFile* ] [**-L** *Library Pathname*]...] [*ObjectFiles*] [*Libraries*]

where:

**-o** *OutPutFile* specifies the name of the generated file (if the **-o** parameter is not present , the *cicsprCOBOL* file is placed in the current directory).

**-L** *Library Pathname* specifies a directory you want to add to the *LIBPATH* in *cicsprCOBOL*.

*ObjectFiles* is the name of the object file you want to use (for example, db2.o).

*Libraries* is the name of the libraries you want to use.

For more detailed information refer to the *AIX CICS/6000 Planning and Installation Guide* and the *AIX CICS/6000 Customization and Operation Guide*.

## A.1 CICS/6000 and DATABASE 2 AIX/6000

To build the MicroFocus COBOL run-time support for CICS/6000 and DATABASE 2 AIX/6000 1.0 you have to run the *cicsmkcobol* with the following options:

```
cicsmkcobol -L /usr/lpp/db2_01_01_0000/lib /usr/lpp/db2_01_01_0000/lib/db2.o
```

Refer to 3.2.2.1, "Shared Object" on page 17 to learn how to create the *db2.o* object file.

## A.2 CICS/6000 and Informix Version 5

Informix Version 5 provides a pair of shared objects, *libinf501cs.o* and *libinf501ci.o*. If you need the ESQL/COBOL preprocessor to be case sensitive with respect to statement identifiers (created with a *PREPARE* statement) and cursor names (created with a *DECLARE* statement), you have to use the *libinf501cs.o* shared object file when you run the *cicsmkcobol* script.

Otherwise you should use the *libinf501ci.o* shared object file. When you install the Informix Version 5 TP/XA product, both libraries are copied in the */usr/lib* directory. The use of the shared object file must be consistent when you create the Switch Load file as documented in 3.3.2.3, "Resource Manager Switch" on page 26.

To build the MicroFocus COBOL run-time support for CICS/6000 and Informix Version 5 you have to run the *cicsmkcobol* script with the following options:

*cicsmkcobol -L /usr/lib /usr/lib/libinf501cs.o*.

## A.3 CICS/6000 and Oracle7

To build the MicroFocus COBOL run-time support for CICS/6000 and Oracle7 you have to run the *cicsmkcobol* script with the following options:

*cicsmkcobol -L $ORACLE_HOME/lib $ORACLE_HOME/procob/lib/cobsqlintf.o $ORACLE_HOME/lib/libsharesqlxa.a*,

where ORACLE_HOME is the installation path of Oracle7 (on our machine, /oracle).

## A.4 CICS/6000 and Sybase System 10

To build the MicroFocus COBOL run-time support for CICS/6000 and Sybase System 10 you have to carry out the following steps:

1. Log in as user root.
2. Set the *COBDIR* environment variable to the directory path for the Micro Focus COBOL installation.
3. Set the *PATH* environment variable to include the Micro Focus COBOL *bin* directory.
4. Change to the proper directory by entering:

   *cd $SYBASE/sample/xalibrary/CICS*.
5. Run *xa_make_cobol_runtime*,

where SYBASE is the installation path of Sybase System 10 (on our machine, /sybase).

**Note**

The *xa_make_cobol_runtime* script runs the *cicsmkcobol*, which is in the */usr/lpp/cics/v1.1/bin* directory. If *cicsmkcobol* is located somewhere else, you must edit the script to reflect the change in location.

The *xa_make_cobol_runtime* script builds a Micro Focus COBOL run-time environment with CICS/6000 and Sybase System 10 XA support. It allows CICS/6000 transactions written in COBOL to reference XA-Library and Open Client functions at run time.

# Appendix B.  Appendix B. Setting Up the UXA1 Demo

In this chapter we describe the steps required to set up the UXA1 demo to work with CICS/6000 and RDBMSs using the XA interface.

We tell you how to create a database demo and table, populate the table, configure the makefile, compile the application, and configure CICS/6000 resources to run the demo with the following relational databases:

- DATABASE 2 AIX/6000
- Informix Version 5
- Oracle7
- Sybase System 10.

## B.1  Before You Start

We assume that your RISC System/6000 is already running DATABASE 2 AIX/6000, Informix Version 5, Oracle7, or Sybase System 10.

AIX 3.2.5 is the minimum operating level required for the CICS/6000 December PTF.

## B.2  DATABASE 2 AIX/6000

The samples are included in the CICS/6000 LPPs.  You can find in the /usr/lpp/cics/v1.1/src/samples/xa directory the files you need to build and run the *UXA1* transaction.  The following files are supplied:

- uxa1.README instructions to build a CICS/6000 C or COBOL transaction
- uxa1.db2 SQL script to create and populate tables
- uxa1.bms CICS/6000 basic mapping support (BMS) source for panels
- uxa1.sqc C source file for DATABASE 2 AIX/6000
- uxa1_db2.mk C makefile for DATABASE 2 AIX/6000.

### B.2.1  Create Database

Log in as a user that owns a DB2/6000 instance and verify that the database is running by issuing the command:

*$ ps -ef | grep db2*

Figure 42 on page 118 shows the output on our machine.

```
   root 12565     1   0   Feb 23      -  0:01 db2licd
    db2 15017 16296   0 15:31:47      -  0:00 db2sysc
    db2 15531 15017   0 15:31:47      -  0:00 Ydb2loggr"
   root 16296     1   0 15:31:47      -  0:00 db2wdog
    db2 16554 15017   0 15:31:47      -  0:00 db2dlock
    db2 24603     1   0   Feb 24      -  0:08 /u/db2/sqllib/bin/db2bp 24
    db2 24875     1   0   Feb 24      -  0:00 /u/db2/sqllib/bin/db2bp 24
    seb 25791 23993   4 16:06:16  pts/0  0:00 grep db2
```

*Figure 42. Processes Related to DATABASE 2 AIX/6000*

If DATABASE 2 AIX/6000 is not running, you can bring it up by entering this command:

*$ db2start*.

Now you are ready to execute the SQL script, uxa1.db2, using the *db2 -Command Line Processor*. The command can be used to execute SQL statements, directly from the operating system command prompt:

*$ db2 -f uxa1.db2*.

The *-f* option tells the command line processor to read the command input from a file instead of from standard input.

Uxa1.db2 creates a *cicstest* database and a table, *cheese*, within the database, populates the table, and grants any user the right to work against the table and connect to the *cicstest* database.

## B.2.2 Run the Makefile

Before running the makefile you must check that the *PATH* environment variable includes the */usr/lpp/cics/v1.1/bin* path. To do this run the command:

*$ echo $PATH*

You also have to check that *DB2INSTANCE* is set in order to identify the instance as required by the DATABASE 2 AIX/6000 configuration (on our machine, *db2loc*). You can do this by running:

*$ echo $DB2INSTANCE*

Now you can enter from the command line:

*$ make -f uxa1_db2.mk*.

Figure 43 on page 119 shows the output on our machine.

```
    cicsmap uxa1.bms
ERZ0511I/0106: Basic Mapping Support (BMS) map translation ended.
0 errors, 0 warnings.
ERZ0513I/0702: Physical map generated to 'uxa1m.map'.
ERZ0514I/0601: Logical map generated to 'uxa1.h'.
    ;db2 connect to cicstest; \
    ;;db2 prep uxa1.sqc; \
    ;;db2 grant execute on package uxa1 to public

       Database Connection Information

     Database product       = DB2/6000 1.1.0
     SQL authorization ID   = DB2
 Local database alias   = CICSTEST


LINE    MESSAGES FOR uxa1.sqc
------  --------------------------------------------------------------------
        SQL0060W  The "C" precompiler is in progress.
        SQL0091W  Precompilation or binding was ended with "0"
                  errors and "0" warnings.
DB20000I  The SQL command completed successfully.
 mv uxa1.c uxa1.ccs
 CCFLAGS="/usr/lib/db2.o"; \
  export CCFLAGS; \
  cicstcl -e -d -lC uxa1.ccs
ERZ0447I/5002: Running the translation step: 'cicstran  -e -d -lC
uxa1.ccs'
ERZ0460I/5015: cicstran translation ended: 0 error(s), 0 warning(s).
ERZ0449I/5003: Running the compile and link step:
'xlc_r /usr/lpp/cics/v1.1/lib/libcics_api.a -e main
-I/usr/lpp/cics/v1.1/lib/libcics_api.a -e main
-I/usr/lpp/cics/v1.1./include /usr/lib/db2.o -o uxa1 uxa1.c'
```

*Figure 43. Output of Makefile to Compile UXA1 Transaction: DATABASE 2 AIX/6000*

The *uxa1_db2.mk* makefile provided with the XA samples assumes that you have run the *db2ln* command to create links for libraries in:

*/usr/lpp/db2_vv_rr_mmmm/lib* to */usr/lib*

and for include files in:

*/usr/lpp/db2_vv_mmmm/include* to */usr/include*.

Refer to the *DATABASE 2 AIX/6000 Installation Guide* for a complete description of the use of the db2ln command.

The makefile creates the *uxa1m.map* map running the *cicsmap* translator, connects to the database, and executes the *prep* command to precompile an application program source file containing embedded SQL statements.

The makefile also creates a package in the database to which a connection has been established, and it grants all users the privilege to execute the package *uxa1*.

As the last step the makefile starts the *cicstcl* translator, which performs the translation, compiles the translated program, and links all resulting objects using the AIX *xlc_r* command to produce the *uxa1* executable program.

Refer to the *AIX CICS/6000 Application Programming Guide* for a complete description of the use of the *cicsmap* and *cicstcl* commands. Refer to *DATABASE 2 AIX/6000 Command Reference* for a complete description of the use of the *prep* and *grant* commands.

**Note**

If you drop the database after the compilation, you have to rerun the *prep* command to create the package. The easiest way to do that is to update the *uxa1.sqc* file by entering:

*$ touch uxa1.sqc*

and then rerunning:

*$ make -f uxa1_db2.mk*

## B.2.3 Configure Programs and Transactions for CICS/6000 Using SMIT for DATABASE 2 AIX/6000

The makefile, as described in B.2.2, “Run the Makefile” on page 118, produces a *uxa1m.map* file and a *uxa1* file that you can move under your CICS/6000 region by entering:

*$ cp uxa1m.map /var/cics_regions/<region>/maps/prime/uxa1m.map*

*$ cp uxa1 /var/cics_regions/<region>/bin/DB2UXA1*.

Your AIX user identifier must belong to the AIX *cics* group to move both files. Renaming the uxa1 program enables you to distinguish it from the name used in a different database.

Now you must define the CICS/6000 resources by using SMIT or writing a shell script that adds all of the entries you need as shown in B.2.4, “Configure Programs and Transactions for CICS/6000 Using Script for DATABASE 2 AIX/6000” on page 123.

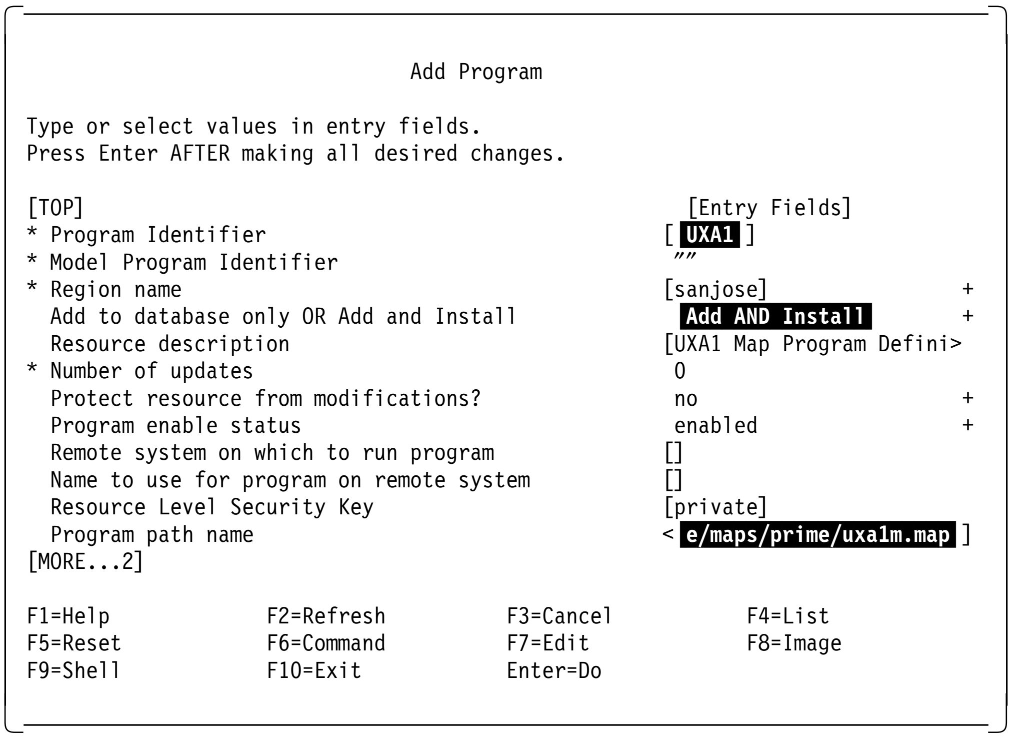
Resource definitions are held in the stanza files, which make up the permanent database that CICS/6000 uses to cold start the system, and in the run-time database, which CICS/6000 uses to hold information about its resources when it is running.

Enter the following command to define a program to the CICS/6000 stanza database:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
      └→ Define Resources for a CICS/6000 Region
         └→ Manage Resource(s)
            └→ Programs
               └→ Add New
                  └→ Model Program Identifier
```

or use the fastpath command, *smitty cicsaddpd*.

When prompted for a model program identifier, choose the model **""**. You will get a panel similar to that shown in Figure 44.



```
                              Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Program Identifier                                  [ UXA1 ]
* Model Program Identifier                             ""
* Region name                                         [sanjose]              +
  Add to database only OR Add and Install              Add AND Install       +
  Resource description                                [UXA1 Map Program Defini>
* Number of updates                                    0
  Protect resource from modifications?                 no                    +
  Program enable status                                enabled               +
  Remote system on which to run program               []
  Name to use for program on remote system            []
  Resource Level Security Key                         [private]
  Program path name                                   < e/maps/prime/uxa1m.map ]
[MORE...2]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

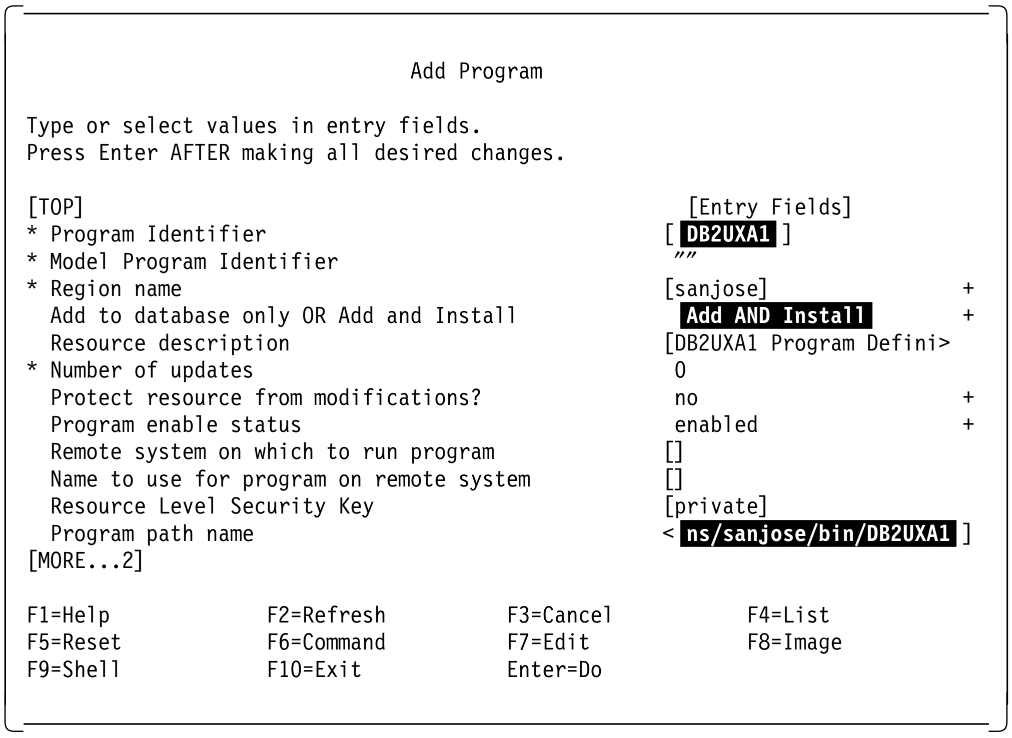
*Figure 44. Defining the UXA1 Map Program to CICS/6000: DATABASE 2 AIX/6000*

The full program path name as shown in Figure 44 is: */var/cics_regions/<region_name>/maps/prime/uxa1m.map*.

**Note**

You must call the UXA1 program identifier; otherwise CICS/6000 returns an APCT error because the application defines a mapset UXA1.

You now have to install the *DB2UXA1* program by running the fastpath command, *smitty cicsaddpd*. You will get the panel shown in Figure 45 on page 122.



```
                                 Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Program Identifier                                  [ DB2UXA1 ]
* Model Program Identifier                             ""
* Region name                                         [sanjose]             +
  Add to database only OR Add and Install              Add AND Install      +
  Resource description                                [DB2UXA1 Program Defini>
* Number of updates                                    0
  Protect resource from modifications?                 no                   +
  Program enable status                                enabled              +
  Remote system on which to run program               []
  Name to use for program on remote system            []
  Resource Level Security Key                         [private]
  Program path name                                   < ns/sanjose/bin/DB2UXA1 ]
[MORE...2]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 45. Defining the DB2UXA1 Program to CICS/6000*

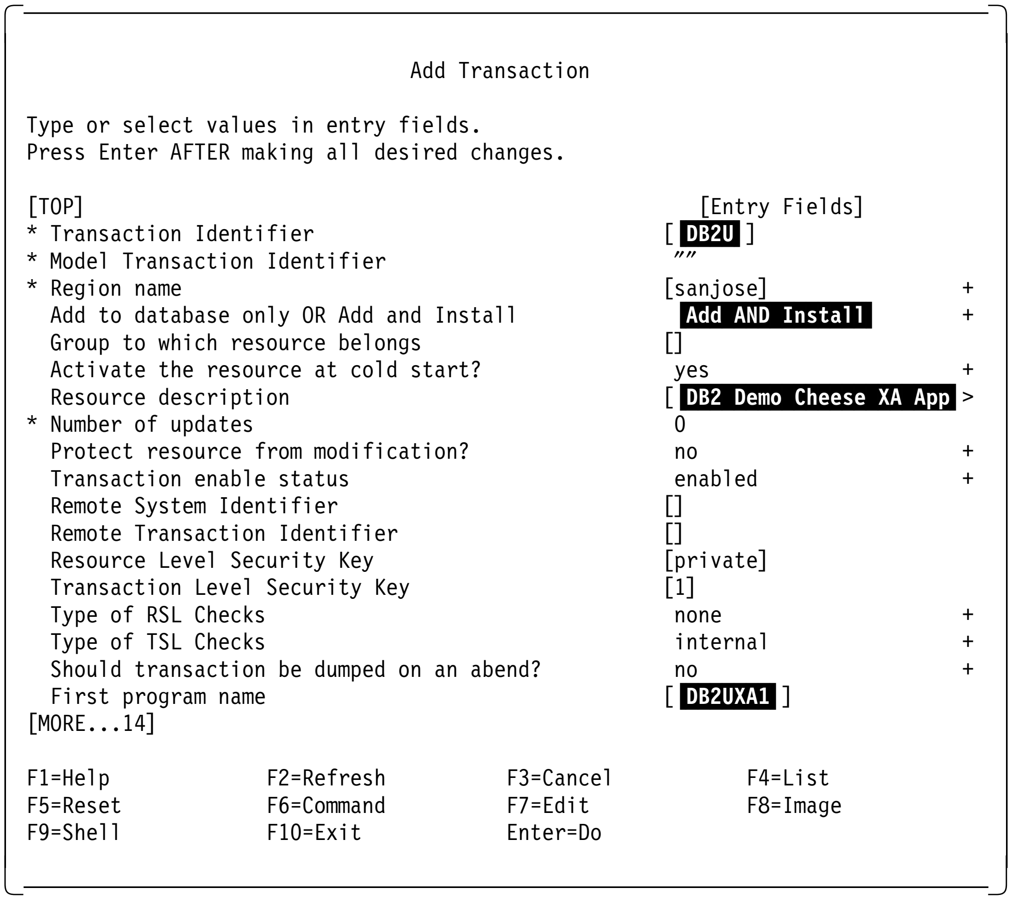
The full program path name as shown in Figure 45 is: */var/cics_regions/<region_name>/bin/DB2UXA1*.

At this point you have to define the *DB2U* transaction to CICS/6000 by entering the following command:

```
$ smitty cics
   └─► Manage CICS/6000 Regions
      └─► Define Resources for a CICS/6000 Region
         └─► Manage Resource(s)
            └─► Transactions
               └─► Add New
                  └─► Model Transaction identifier
```

or by using the fastpath command, *smitty cicsaddtd*.

When prompted for a model transaction identifier, choose the model **""**. You will get a panel similar to that shown in Figure 46 on page 123.



```
                              Add Transaction

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Transaction Identifier                            [ DB2U ]
* Model Transaction Identifier                       ""
* Region name                                       [sanjose]               +
  Add to database only OR Add and Install            Add AND Install        +
  Group to which resource belongs                   []
  Activate the resource at cold start?               yes                    +
  Resource description                              [ DB2 Demo Cheese XA App >
* Number of updates                                  0
  Protect resource from modification?                no                     +
  Transaction enable status                          enabled                +
  Remote System Identifier                          []
  Remote Transaction Identifier                     []
  Resource Level Security Key                       [private]
  Transaction Level Security Key                    [1]
  Type of RSL Checks                                 none                   +
  Type of TSL Checks                                 internal               +
  Should transaction be dumped on an abend?          no                     +
  First program name                                [ DB2UXA1 ]
[MORE...14]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 46. Defining the DB2U Transaction to CICS/6000*

## B.2.4 Configure Programs and Transactions for CICS/6000 Using Script for DATABASE 2 AIX/6000

If you prefer, you can configure programs and transactions by writing a shell script as shown in Figure 47 on page 124.

```
#!/bin/ksh
echo $CICSREGION
BINDIR=/var/cics_regions/$CICSREGION/bin
MAPDIR=/var/cics_regions/$CICSREGION/maps/prime

cp uxa1 $BINDIR/DB2UXA1
cp uxa1m.map $MAPDIR

cicsadd -c td -r $CICSREGION \
                ResourceDescription=″DB2 Demo Cheese XA Application″ \
                ProgName=″DB2UXA1″ \
                DB2U

cicsadd -c pd -r $CICSREGION -P ″DB2UXA1″ \
                ResourceDescription=″DB2UXA1 Program Definition″ \
                PathName=″$BINDIR/DB2UXA1″

cicsadd -c pd -r $CICSREGION -P ″UXA1″ \
                ResourceDescription=″UXA1 Map Definitions″ \
                PathName=″$MAPDIR/uxa1m.map″ \
                ProgType=map
```

*Figure 47. Sample Shell Script to Configure CICS/6000 Resources: DATABASE 2 AIX/6000*

These configurations take effect only after a cold start of CICS/6000 because we have defined resources only to database stanza files.

## B.3 Informix Version 5

The samples are included in CICS/6000 LPPs, and you can find in the /usr/lpp/cics/v1.1/src/samples/xa directory the files you need to build and run the *UXA1* transaction. The following files are supplied:

- uxa1.README instructions to build a CICS/6000 C or COBOL transaction
- uxa1.sql SQL script to create and populate tables
- uxa1.bms CICS/6000 BMS source for panels
- uxa1.ec C source file for Informix
- uxa1_inf5.mk C Makefile for Informix.

**Note**

The */usr/lpp/cics/v1.1/doc* directory explains how to configure Informix Version 5 with CICS/6000. Please read *informix501.README* file.

### B.3.1 Create Database

Login as user Informix and verify that the database is running by using the command:

*$ ps -ef | grep informix.*

Figure 48 on page 125 shows the output on our machine.

```
informix  8130     1   0   Feb 28  hft/0  2:41 tbinit -isy
informix 19139  8130   0   Feb 28  hft/0  0:03 tbpgcl
informix 22879  8130   0 17:22:08  hft/0  0:00 tbundo
     seb 23832 26899   6 11:14:22  pts/0  0:00 grep informix
```

Figure 48. Processes Related to Informix

If Informix is not running, you can bring it up by entering the command:

*$ tbinit*.

Use the *ps* command to check that the *tbinit* and *tbpgcl* daemons are running. We created a *cicstest* database with the *DB-Access* utility by using the command:

*$ dbaccess -dc cicstest*,

and we changed the logging from buffered to unbuffered, so, in the event of a failure, only the single alteration in progress at the time of the failure is lost. You can do that by running this command:

*$ tbtape -s -U cicstest*.

CICS/6000 expects to work with unbuffered logging.

You will be asked to perform an archive. Type 0 and you will get the messages shown in Figure 49.

```
Please enter the level of archive to be performed (0, 1, or 2)


Please label this tape as number 1 in the archive sequence.

Level 0 archive is 100 percent completed.

date:  Fri Mar  4 09:45:38 1994

Tapes required to restore the system to the state
at the beginning of this archive:

Archive level:  0     Archive date:  Fri Mar  4 09:45:38 1994

Logical log unique id at the beginning of the archive:  1

Database(s) logging status successfully updated.


Program over.
```

Figure 49. Results of tbtape Command

Now you must grant to public the *cicstest* database; we suggest that you add this line to the uxa1.sql file:

*grant resource to public;*

Furthermore the uxa1.sql file contains some lines that begin with a pound sign (#) that dbaccess cannot understand, so you have to add the right delimiters for comments { } or delete those lines.

Now you are ready to execute the SQL uxa1.sql script by typing:

*$ dbaccess cicstest uxa1*.

Uxa1.sql creates a table, *cheese*, within the cicstest database, populates the table, and grants any user the right to work against the table and connect to the *cicstest* database.

## B.3.2 Run the Makefile

Before running the makefile you must edit it to reflect your Informix Version 5 customization. The makefile assumes that INFORMIXDIR is set to */usr/informix*; we modified the uxa1_inf5.mk as shown in Figure 50.

```
INFORMIXDIR=/home/informix

all: uxa1m.map uxa1

uxa1m.map: uxa1.bms
    cicsmap uxa1.bms

uxa1: uxa1.ec
    $(INFORMIXDIR)/bin/esql -e uxa1.ec
    mv uxa1.c uxa1.ccs
    CCFLAGS="-I$(INFORMIXDIR)/incl/esql \
          -L/home/informix/lib/esql \
          /home/informix/lib/esql/libinf501cs.o"; \
    export CCFLAGS; \
    cicstcl -e -d -lC uxa1.ccs
```

*Figure 50. uxa1_inf5.mk Makefile after Editing*

Now you can enter from the command line:

*$ make -f uxa1_inf5.mk.*

Figure 51 on page 127 shows the output on our machine.

```
    cicsmap uxa1.bms
ERZ0511I/0106: Basic Mapping Support (BMS) map translation ended.
0 errors, 0 warnings.
ERZ0513I/0702: Physical map generated to 'uxa1m.map'.
ERZ0514I/0601: Logical map generated to 'uxa1.h'.
    /home/informix/bin/esql -e uxa1.ec
    mv uxa1.c uxa1.ccs
    CCFLAGS="-I/home/informix/incl/esql \
              -L/home/informix/lib/esql \
              /home/informix/lib/esql/libinf501cs.o"; \
          export CCFLAGS; \
          cicstcl -e -d -lC uxa1.ccs
ERZ0447I/5002: Running the translation step: 'cicstran  -e -d -lC uxa1.ccs'
ERZ0460I/5015: cicstran translation ended: 0 error(s), 0 warning(s).
ERZ0449I/5003: Running the compile and link step:
'xlc_r /usr/lpp/cics/v1.1/lib/libcics_api.a -e main
-I/usr/lpp/cics/v1.1/include -I/home/informix/incl/esql
-L/home/informix/lib/esql /home/informix/lib/esql/libinf501cs.o -o uxa1 uxa1.c'
```

*Figure 51. Output of uxa1_inf5.mk Makefile to Compile UXA1 Transaction: Informix Version 5*

Refer to the *Informix - OnLine Administrator's Guide* for more details about how to use Informix utilities.

### B.3.3 Configure Programs and Transactions for CICS/6000 Using SMIT for Informix Version 5

The makefile, as described in B.3.2, "Run the Makefile" on page 126, produces a *uxa1m.map* file and a *uxa1* file that you can move under your CICS/6000 region by entering:

*$ cp uxa1m.map /var/cics_regions/<region>/maps/prime/uxa1m.map*

*$ cp uxa1 /var/cics_regions/<region>/bin/INFUXA1.*

Your AIX user identifier must belong to the AIX *cics* group to move both files. Renaming the uxa1 program enables you to distinguish it from the name used in a different database.

Now you must define the CICS/6000 resources by using SMIT or writing a shell script that adds all of the entries you need as shown in B.3.4, "Configure Programs and Transactions for CICS/6000 Using Script for Informix Version 5" on page 130.

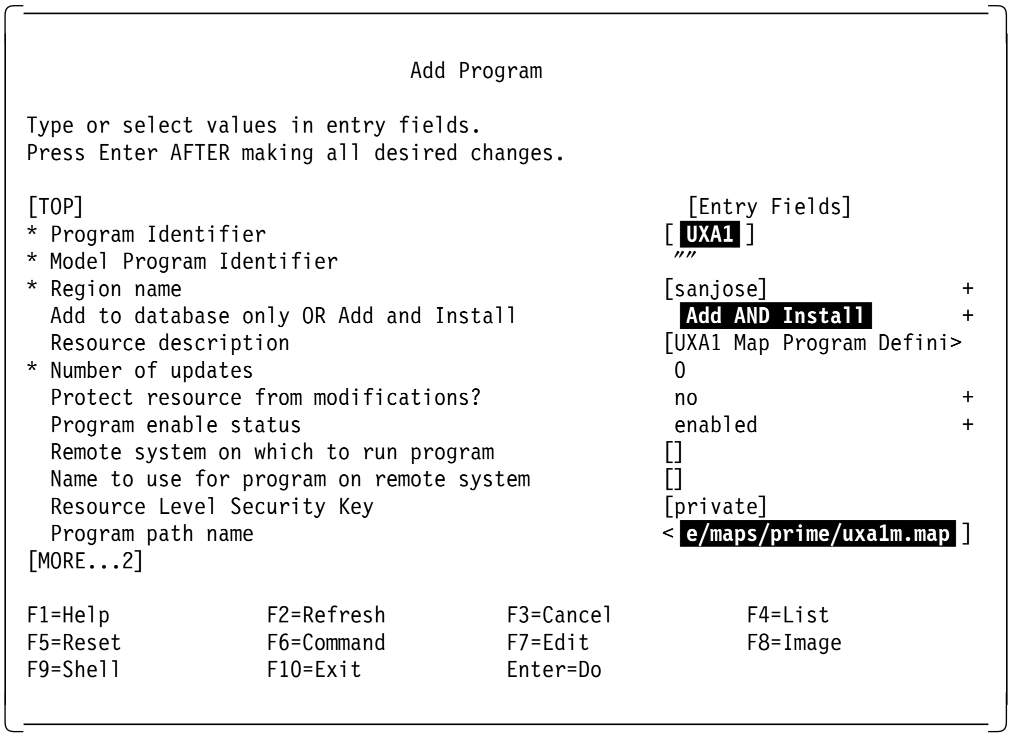
Resource definitions are held in the stanza files, which make up the permanent database that CICS/6000 uses to cold start the system, and in the run-time database, which CICS/6000 uses to hold information about its resources when it is running.

Enter the following command to define a program to the CICS/6000 stanza database:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
       └→ Define Resources for a CICS/6000 Region
           └→ Manage Resource(s)
               └→ Programs
                   └→ Add New
                       └→ Model Program Identifier
```

or use the fastpath command, *smitty cicsaddpd*.

When prompted for a model program identifier, choose the model **""**. You will get a panel similar to that shown in Figure 52



```
                                Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Program Identifier                                 [ UXA1 ]
* Model Program Identifier                            ""
* Region name                                        [sanjose]           +
  Add to database only OR Add and Install             Add AND Install    +
  Resource description                               [UXA1 Map Program Defini>
* Number of updates                                   0
  Protect resource from modifications?                no                 +
  Program enable status                               enabled            +
  Remote system on which to run program              []
  Name to use for program on remote system           []
  Resource Level Security Key                        [private]
  Program path name                                  < e/maps/prime/uxa1m.map ]
[MORE...2]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

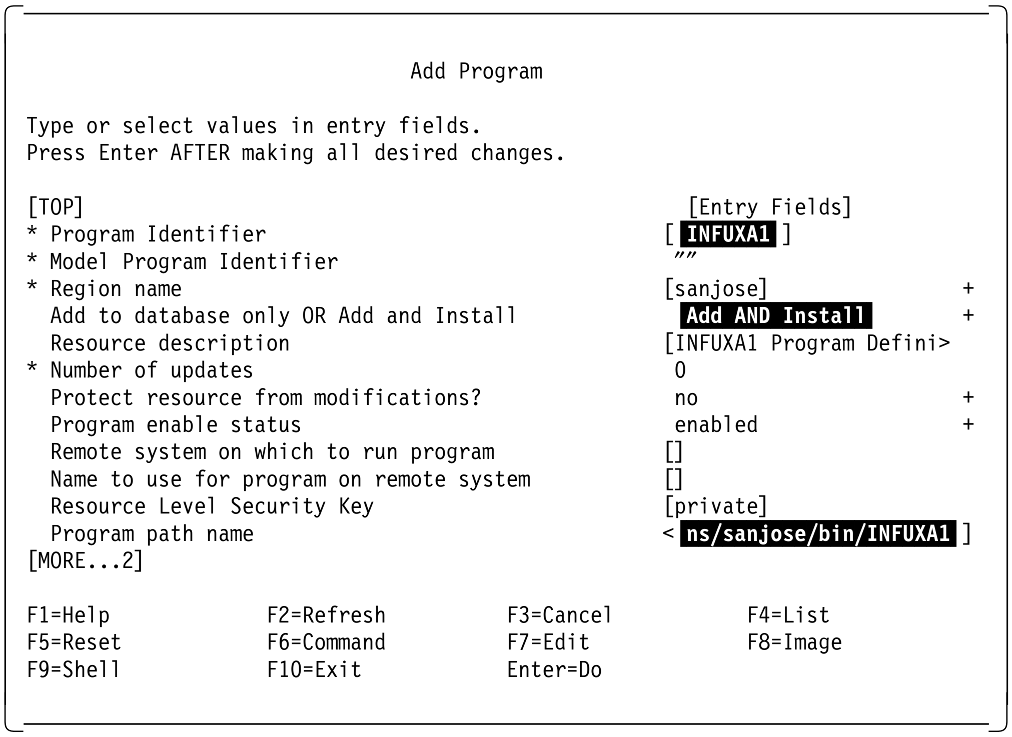
*Figure 52. Defining the UXA1 Map Program to CICS/6000: Informix Version 5*

The full program path name as shown in Figure 52 is: */var/cics_regions/<region_name>/maps/prime/uxa1m.map*.

**Note**

You must call the UXA1 program identifier; otherwise CICS/6000 returns an APCT error because the application defines a mapset UXA1.

You now have to install the *INFUXA1* program by running the fastpath command, *smitty cicsaddpd*. You will get the panel shown in Figure 53 on page 129.



```
                              Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                [Entry Fields]
* Program Identifier                                [ INFUXA1 ]
* Model Program Identifier                           ""
* Region name                                       [sanjose]             +
  Add to database only OR Add and Install             Add AND Install     +
  Resource description                              [INFUXA1 Program Defini>
* Number of updates                                  0
  Protect resource from modifications?               no                   +
  Program enable status                              enabled              +
  Remote system on which to run program             []
  Name to use for program on remote system          []
  Resource Level Security Key                       [private]
  Program path name                                 < ns/sanjose/bin/INFUXA1 ]
[MORE...2]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 53. Defining the INFUXA1 Program to CICS/6000*

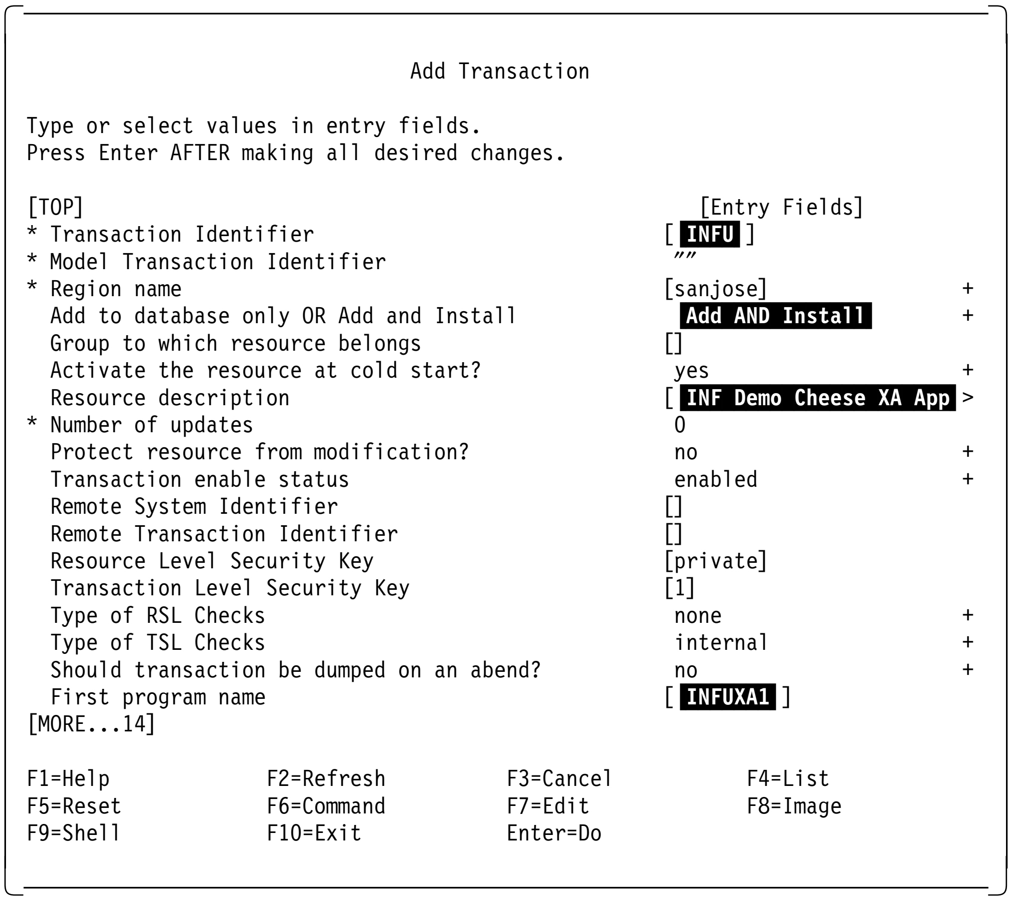
The full program path name as shown in Figure 45 on page 122 is: */var/cics_regions/<region_name>/bin/INFUXA1*.

At this point you have to define the *INFU* transaction to CICS/6000 by entering the following command:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
      └→ Define Resources for a CICS/6000 Region
         └→ Manage Resource(s)
            └→ Transactions
               └→ Add New
                  └→ Model Transaction identifier
```

or by using the fastpath command: *smitty cicsaddtd*.

When prompted for a model transaction identifier, choose the model **""**. You will get a panel similar to that shown in Figure 54 on page 130.



```
                            Add Transaction

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
* Transaction Identifier                               [ INFU ]
* Model Transaction Identifier                          ""
* Region name                                          [sanjose]               +
  Add to database only OR Add and Install                Add AND Install       +
  Group to which resource belongs                      []
  Activate the resource at cold start?                  yes                    +
  Resource description                                 [ INF Demo Cheese XA App >
* Number of updates                                     0
  Protect resource from modification?                   no                     +
  Transaction enable status                             enabled                +
  Remote System Identifier                             []
  Remote Transaction Identifier                        []
  Resource Level Security Key                          [private]
  Transaction Level Security Key                       [1]
  Type of RSL Checks                                    none                   +
  Type of TSL Checks                                    internal               +
  Should transaction be dumped on an abend?             no                     +
  First program name                                   [ INFUXA1 ]
[MORE...14]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 54. Defining the INFU Transaction to CICS/6000*

## B.3.4 Configure Programs and Transactions for CICS/6000 Using Script for Informix Version 5

You can configure programs and transaction by writing a shell script as shown in Figure 55 on page 131.

```
#!/bin/ksh
echo $CICSREGION
BINDIR=/var/cics_regions/$CICSREGION/bin
MAPDIR=/var/cics_regions/$CICSREGION/maps/prime

cp uxa1 $BINDIR/INFUXA1
cp uxa1m.map $MAPDIR

cicsadd -c td -r $CICSREGION \
                ResourceDescription=″INF Demo Cheese XA Application″ \
                ProgName=″INFUXA1″ \
                INFU

cicsadd -c pd -r $CICSREGION -P ″INFUXA1″ \
                ResourceDescription=″INFUXA1 Program Definition″ \
                PathName=″$BINDIR/INFUXA1″

cicsadd -c pd -r $CICSREGION -P ″UXA1″ \
                ResourceDescription=″UXA1 Map Definitions″ \
                PathName=″$MAPDIR/uxa1m.map″ \
                ProgType=map
```

*Figure 55. Sample Shell Script to Configure CICS/6000 Resources: Informix Version 5*

These configurations take effect only after a cold start of CICS/6000 because we have defined resources only to database stanza files.

## B.4 Oracle7

The samples are included in CICS/6000 LPPs, and you can find in the /usr/lpp/cics/v1.1/src/samples/xa directory the files you need to build and run the *UXA1* transaction. The following files are supplied:

- uxa1.README instructions to build a CICS/6000 C or COBOL transaction
- uxa1.sql SQL script to create and populate tables
- uxa1.bms CICS/6000 BMS source for panels
- uxa1.ec C source file for Informix
- uxa1_inf5.mk C Makefile for Informix.

**Note**

The */usr/lpp/cics/v1.1/doc* directory explains how to configure Informix Version 5 with CICS/6000. Please read the *oracle713.README* file.

### B.4.1 Create Database

Log in as user Oracle and verify that the database is running by using the command:

*$ ps -ef | grep oracle*.

Figure 56 on page 132 shows the output on our machine.

```
   seb  8846 20353   5 11:38:10  pts/7  0:00 grep oracle
oracle 12359     1   0 10:54:27      -  0:01 ora_dbwr_ora7
oracle 13640     1   0 10:54:29      -  0:00 ora_lgwr_ora7
oracle 18758     1   0 10:54:25      -  0:00 ora_pmon_ora7
oracle 19275     1   0 10:54:35      -  0:00 ora_d000_ora7
oracle 21321     1   0 10:54:31      -  0:01 ora_smon_ora7
oracle 21578     1   0 10:54:33      -  0:00 ora_s000_ora7
```

*Figure 56. Processes Related to Oracle*

If Oracle is not running, you can bring it up by entering the command:

*$ sqldba lmode=yes*.

You will be prompted with:

*SQLDBA>*.

Type:

*SQLDBA> connect internal*

*SQLDBA> startup*

*SQLDBA> exit*.

Use the *ps* command to check that the Oracle processes are running.

The uxa1.sql file contains some lines that begin with a pound sign (#) that Oracle′s utility *SQLPLUS* cannot understand. You can either add the right delimiters for comments { } or delete those lines.

Now you are ready to execute the SQL uxa1.sql script by typing:

*$ sqlplus scott/tiger @uxa1.sql*.

**Note**

The user *scott/tiger* is available if you install the database demo during Oracle installation; otherwise you have to create an Oracle user.

Uxa1.sql creates a table, *cheese*, populates the table, and grants any user the right to work against the table.

### B.4.2 Run the Makefile

Before running the makefile you must edit it to reflect your Oracle7 customization. The makefile assumes that ORACLE_HOME is set to */usr/oracle*; we modified the uxa1_ora7.mk as shown in Figure 57 on page 133.

```
ORACLE_HOME=/home/oracle

all: uxa1m.map uxa1

uxa1m.map: uxa1.bms
    cicsmap uxa1.bms

uxa1: uxa1.pc
    $(ORACLE_HOME)/bin/proc release_cursor=yes sqlcheck=none \
        ireclen=512 iname=uxa1.pc
    mv uxa1.c uxa1.ccs
    CCFLAGS="-L/home/oracle/lib \
          -lsharesqlxa -lm -lld"; \
    export CCFLAGS; \
    cicstcl -e -d -lC uxa1.ccs
    rm -f uxa1.c
```

*Figure 57. uxa1_ora7.mk Makefile after Editing*

Now you can enter from the command line:

*$ make -f uxa1_ora7.mk.*

Figure 58 shows the output on our machine.

```
    cicsmap uxa1.bms
ERZ0511I/0106: Basic Mapping Support (BMS) map translation ended.
               0 errors, 0 warnings.
ERZ0513I/0702: Physical map generated to 'uxa1m.map'.
ERZ0514I/0601: Logical map generated to 'uxa1.h'.
    /home/oracle/bin/proc release_cursor=yes sqlcheck=none
ireclen=512 iname=uxa1.pc

Pro*C: Release 1.5.9.0.1 - Production on Tue Mar  8 16:54:01 1994

Copyright (c) Oracle Corporation 1979, 1992.  All rights reserved.


Precompiling uxa1.pc
    mv uxa1.c uxa1.ccs
    CCFLAGS="-L/home/oracle/lib \
               -lsharesqlxa -lm -lld"; \
          export CCFLAGS; \
          cicstcl -e -d -lC uxa1.ccs
ERZ0447I/5002: Running the translation step: 'cicstran  -e -d -lC uxa1.ccs'
ERZ0460I/5015: cicstran translation ended: 0 error(s), 0 warning(s).
ERZ0449I/5003: Running the compile and link step:
'xlc_r /usr/lpp/cics/v1.1/lib/libcics_api.a -e main -I/usr/lpp/cics/v1.1/include
-L/home/oracle/lib -lsharesqlxa -lm -lld -o uxa1 uxa1.c'
"uxa1.c", line 345.22: 1506-193 (E) Function call argument cannot be assigned
to corresponding parameter.
    rm -f uxa1.c
```

*Figure 58. Output of uxa1_ora7.mk Makefile to Compile UXA1 Transaction*

**Note**

We received error 1506-193 (see Figure 59 on page 135) because of the incorrect use of the *sprintf* function library:

*sprintf(name.arr, "%s", panel1.panel1i.newnamei);*

A possible workaround is this:

*sprintf(temp_name, "%s", panel1.panel1i.newnamei);*

*strcpy(name.arr,temp_name);*

where *temp_name* is defined as:

*char temp_name[16];*

Refer to the *Oracle7 Server Administrator's Guide* for more details about how to use Oracle utilities.

### B.4.3 Configure Programs and Transactions for CICS/6000 Using SMIT for Oracle7

The makefile, as described in B.4.2, "Run the Makefile" on page 132, produces a *uxa1m.map* file and a *uxa1* file that you can move under your CICS/6000 region by entering:

*$ cp uxa1m.map /var/cics_regions/<region>/maps/prime/uxa1m.map*

*$ cp uxa1 /var/cics_regions/<region>/bin/ORAUXA1*

Your AIX user identifier must belong to the AIX *cics* group to move both files. Renaming the uxa1 program enables you to distinguish it from the name used in a different database.

Now you must define the CICS/6000 resources by using SMIT or writing a shell script that adds all of the entries you need as shown in B.4.4, "Configure Programs and Transactions for CICS/6000 Using Script for Oracle7" on page 137.

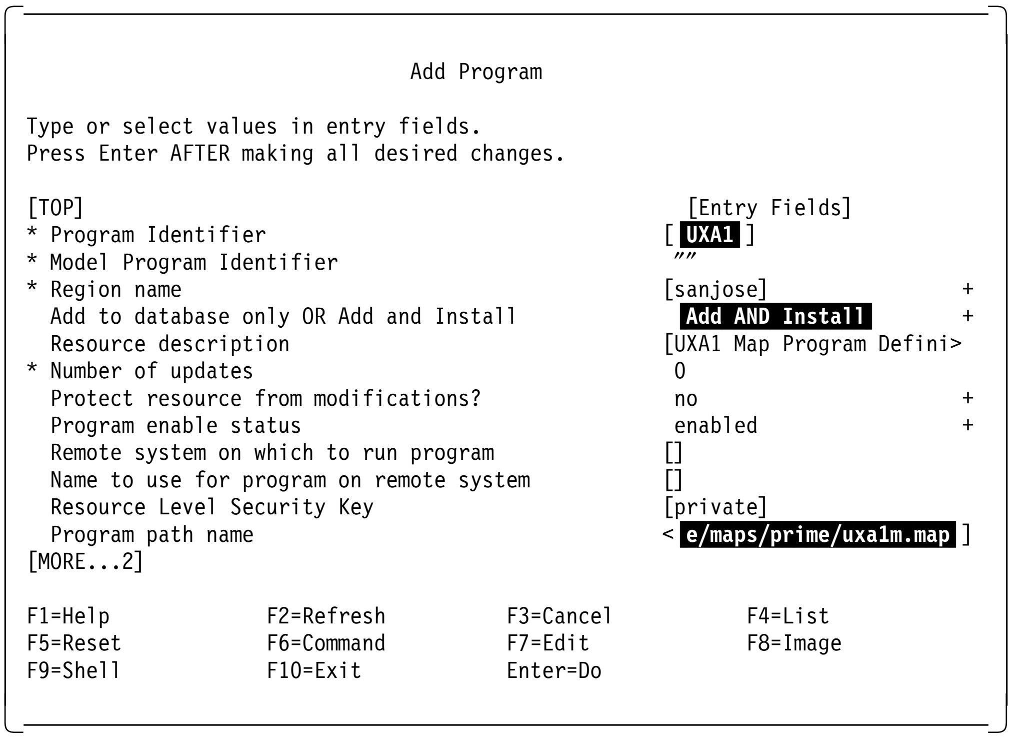
Resource definitions are held in the stanza files, which make up the permanent database that CICS/6000 uses to cold start the system, and in the run-time database, which CICS/6000 uses to hold information about its resources when it is running.

Enter the following command to define a program to the CICS/6000 stanza database:

```
$ smitty cics
  └─► Manage CICS/6000 Regions
      └─► Define Resources for a CICS/6000 Region
          └─► Manage Resource(s)
              └─► Programs
                  └─► Add New
                      └─► Model Program Identifier
```

or use the fastpath command, *smitty cicsaddpd*.

When prompted for a model program identifier, choose the model ″″. You will get a panel similar to that shown in Figure 59



```
                                Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Program Identifier                                 [ UXA1 ]
* Model Program Identifier                            ""
* Region name                                        [sanjose]             +
  Add to database only OR Add and Install             Add AND Install      +
  Resource description                               [UXA1 Map Program Defini>
* Number of updates                                   0
  Protect resource from modifications?                no                   +
  Program enable status                               enabled              +
  Remote system on which to run program              []
  Name to use for program on remote system           []
  Resource Level Security Key                        [private]
  Program path name                                  < e/maps/prime/uxa1m.map ]
[MORE...2]

F1=Help            F2=Refresh          F3=Cancel           F4=List
F5=Reset           F6=Command          F7=Edit             F8=Image
F9=Shell           F10=Exit            Enter=Do
```

*Figure 59. Defining the UXA1 Map Program to CICS/6000: Oracle7*

The full program path name as shown in Figure 59 is: */var/cics_regions/<region_name>/maps/prime/uxa1m.map*.

**Note**

You must call the UXA1 program identifier; otherwise CICS/6000 returns an APCT error because the application defines a mapset UXA1.

You now have to install the *ORAUXA1* program by running the fastpath command, *smitty cicsaddpd*. You will get the panel shown in Figure 60 on page 136.

```
                              Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Program Identifier                                 [ ORAUXA1 ]
* Model Program Identifier                            ""
* Region name                                        [sanjose]          +
  Add to database only OR Add and Install             Add AND Install   +
  Resource description                               [ORAUXA1 Program Defini>
* Number of updates                                   0
  Protect resource from modifications?                no                +
  Program enable status                               enabled           +
  Remote system on which to run program              []
  Name to use for program on remote system           []
  Resource Level Security Key                        [private]
  Program path name                                  < ns/sanjose/bin/ORAUXA1 ]
[MORE...2]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 60. Defining the ORAUXA1 Program to CICS/6000*

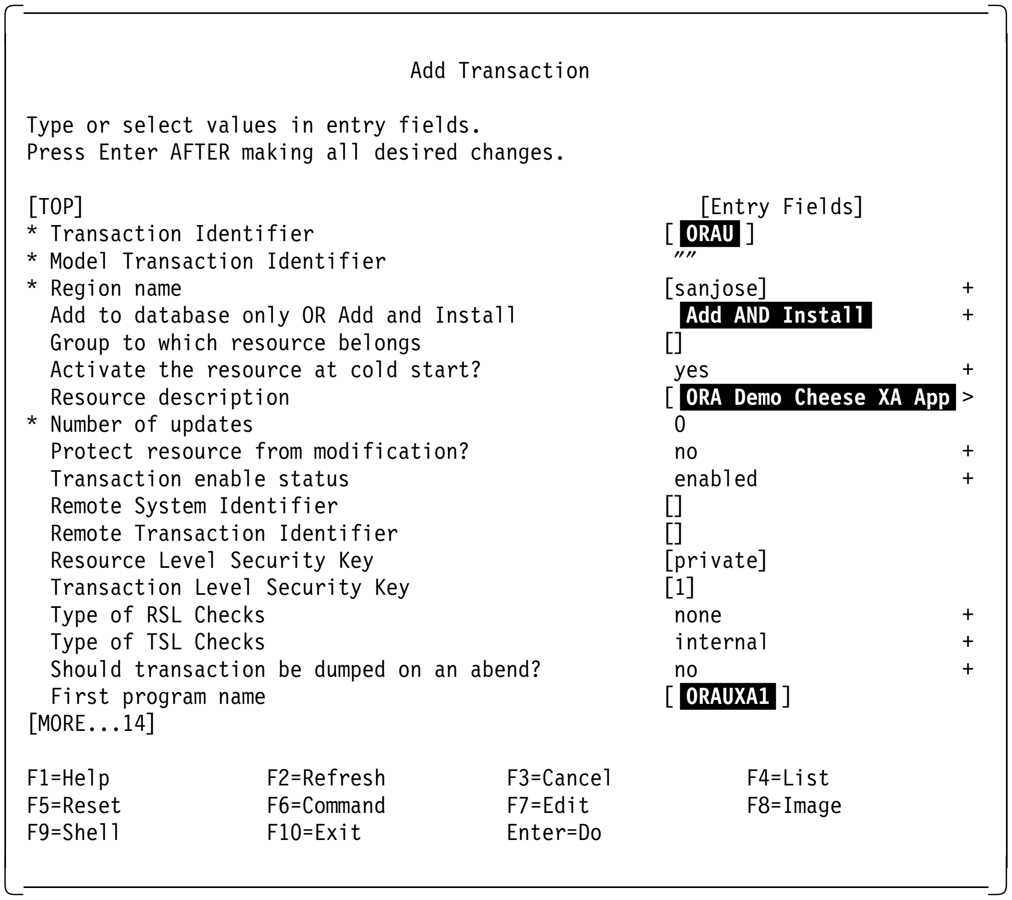
The full program path name as shown in Figure 45 on page 122 is: */var/cics_regions/<region_name>/bin/ORAUXA1*.

At this point you have to define the *ORAU* transaction to CICS/6000 by entering the following command:

```
$ smitty cics
   └─► Manage CICS/6000 Regions
      └─► Define Resources for a CICS/6000 Region
         └─► Manage Resource(s)
            └─► Transactions
               └─► Add New
                  └─► Model Transaction identifier
```

or by using the fastpath command: *smitty cicsaddtd*.

When prompted for a model transaction identifier, choose the model "". You will get a panel similar to that shown in Figure 61 on page 137.



```
                           Add Transaction

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                    [Entry Fields]
* Transaction Identifier                              [ ORAU ]
* Model Transaction Identifier                         ""
* Region name                                         [sanjose]            +
  Add to database only OR Add and Install              Add AND Install     +
  Group to which resource belongs                     []
  Activate the resource at cold start?                 yes                 +
  Resource description                                [ ORA Demo Cheese XA App >
* Number of updates                                    0
  Protect resource from modification?                  no                  +
  Transaction enable status                            enabled             +
  Remote System Identifier                            []
  Remote Transaction Identifier                       []
  Resource Level Security Key                         [private]
  Transaction Level Security Key                      [1]
  Type of RSL Checks                                   none                +
  Type of TSL Checks                                   internal            +
  Should transaction be dumped on an abend?            no                  +
  First program name                                  [ ORAUXA1 ]
[MORE...14]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 61. Defining the ORAU Transaction to CICS/6000*

## B.4.4 Configure Programs and Transactions for CICS/6000 Using Script for Oracle7

You can configure programs and transaction by writing a shell script as shown in Figure 62 on page 138.

```
#!/bin/ksh
echo $CICSREGION
BINDIR=/var/cics_regions/$CICSREGION/bin
MAPDIR=/var/cics_regions/$CICSREGION/maps/prime

cp uxa1 $BINDIR/ORAUXA1
cp uxa1m.map $MAPDIR

cicsadd -c td -r $CICSREGION \
                ResourceDescription="ORA Demo Cheese XA Application" \
                ProgName="ORAUXA1" \
                ORAU

cicsadd -c pd -r $CICSREGION -P "ORAUXA1" \
                ResourceDescription="ORAUXA1 Program Definition" \
                PathName="$BINDIR/ORAUXA1"

cicsadd -c pd -r $CICSREGION -P "UXA1" \
                ResourceDescription="UXA1 Map Definitions" \
                PathName="$MAPDIR/uxa1m.map" \
                ProgType=map
```

*Figure 62. Sample Shell Script to Configure CICS/6000 Resources: Oracle7*

These configurations take effect only after a cold start of CICS/6000 because we have defined resources only to database stanza files.

## B.5 Sybase System 10

We had the demo, *Cheese*, with the beta code of the Sybase XA-Library for CICS/6000. The following files are supplied:

- uxa1.sql SQL script to create and populate tables
- uxa1.cpre C source file for Sybase
- uxa1_syb.mk makefile for Sybase.

### B.5.1 Create Database

Log in as user Sybase and verify that the database is running by using the command:

*$ ps -ef | grep syabse.*

Figure 63 shows the output on our machine.

```
  sybase  9544 11847  10 10:36:15  hft/2  0:44 /sybase/bin/dataserver
   -d/dev/rmasterlv -sSYBASE -e/sybase/install/errorlog -i/s
  sybase 11847     1   0 10:36:15  hft/2  0:00 sh ./RUN_SYBASE
     seb 14949 14689   3 10:54:21  pts/0  0:00 grep sybase
```

*Figure 63. Processes Releted to Sybase*

If Sybase is not running, you can bring it up by entering the command:

*$ startserver*,

which you can find in the *$SYBASE/install* directory; use the *ps* command to check that the Sybase processes are running.

Execute the SQL script uxa1.sql by typing:

*$ isql -Usa -P<Server password> -i uxa1.sql*.

**Note**

The user, *sa*, is available after the Sybase System 10 Server installation, but with a null password. You are required to set a password for the System Administrator account by using the *sp_password* system procedure as documented in the *Sybase SQL Server Installation Guide for IBM RISC System/6000*. The installation process adds a pseudo-user, ″probe″, for use by the two-phase commit program only.

Uxa1.sql creates a table, *cheese*, populates the table, and grants any user the right to work against the table.

## B.5.2 Run the Makefile

The makefile requires that you modify your *SYBASE* environment variable to reflect your Sybase System 10 installation (see Figure 64).

```
SHAROBJS = -lcs.so -lcomn.so

SYB_SHAR_LIBDIR = $(SYBASE)/lib

LIBDIR = $(SYBASE)/lib

SYBLIBS = -lct -lintl -ltcl -lm -linsck
CFLAGS = -c -g
all: uxa1m.map uxa1

uxa1m.map: uxa1.bms
    cicsmap uxa1.bms

uxa1: uxa1.cpre
    $(SYBASE)/bin/cpre uxa1.cpre
    mv uxa1.c uxa1.ccs
    CCFLAGS=″-I $(CICSPATH)/v1.1/include -I $(SYBASE)/include \
    -L $(SYB_SHAR_LIBDIR)\
    -L $(LIBDIR) \
    $(SHAROBJS) \
    $(SYBLIBS)″; \
    export CCFLAGS; \
    cicstcl -e -d -lC uxa1.ccs
```

*Figure 64. uxa1_syb.mk Makefile*

To compile the demo enter from the command line:

*$ make -f uxa1_syb.mk*

Figure 65 on page 140 shows the output on our machine.

```
    cicsmap uxa1.bms
ERZ0511I/0106: Basic Mapping Support (BMS) map translation ended.
               0 errors, 0 warnings.
ERZ0513I/0702: Physical map generated to 'uxa1m.map'.
ERZ0514I/0601: Logical map generated to 'uxa1.h'.
    /sybase/bin/cpre uxa1.cpre
M_WHEN_WARN,Unable to find the SQL statement 'WHENEVER WARNING'.
M_WHEN_NF,Unable to find the SQL statement 'WHENEVER NOT FOUND'.
0 Error(s) and 2 Warning(s) found.
Statistical Report:
    Program name: cpre
    Options specified:
    Input file name: uxa1.cpre
    Listing file name:
    Target file name: uxa1.c
    ISQL file name:
    Tag ID specified:
    Compiler used: ANSI_C
    Open Client version: CS_VERSION_100
    Number of information messages: 11
    Number of warning messages: 2
    Number of error messages: 0
    Number of SQL statements parsed: 6
    Number of host variables declared: 5
    Number of SQL cursors declared: 0
    Number of dynamic SQL statements: 0
    Number of stored Procedures generated: 0
    Connection(s) information:
          User id:
          Server:
          Database:
    mv uxa1.c uxa1.ccs
    CCFLAGS="-I /v1.1/include -I /sybase/include \
          -L /sybase/lib\
          -L /sybase/lib \
          -lcs.so -lcomn.so  \
          -lct -lintl -ltcl -lm -linsck"; \
          export CCFLAGS; \
          cicstcl -e -d -lC uxa1.ccs
ERZ0447I/5002: Running the translation step: 'cicstran  -e -d -lC uxa1.ccs'
ERZ0460I/5015: cicstran translation ended: 0 error(s), 0 warning(s).
ERZ0449I/5003: Running the compile and link step:
'xlc_r /usr/lpp/cics/v1.1/lib/libcics_api.a -e main -I/usr/lpp/cics/v1.1/include
-I /v1.1/include -I /sybase/include -L /sybase/lib -L /sybase/lib -lcs.so
-lcomn.so -lct -lintl -ltcl -lm -linsck -o uxa1 uxa1.c'
```

*Figure 65. Output of uxa1_syb.mk Makefile to Compile UXA1 Transaction*

**Note**

In the *uxa1.cpre* we added an SQL statement to connect to the server defined in the *$SYBASE/xa_config* file; otherwise the application cannot manage any data. The statement is:

*EXEC SQL SET CONNECTION connection_1;*

## B.5.3 Configure Programs and Transactions for CICS/6000 Using SMIT for Sybase System 10

The makefile, as described in B.5.2, “Run the Makefile” on page 139, produces a *uxa1m.map* file and a *uxa1* file that you can move under your CICS/6000 region by entering:

*$ cp uxa1m.map /var/cics_regions/<region>/maps/prime/uxa1m.map*

*$ cp uxa1 /var/cics_regions/<region>/bin/SYBUXA1*.

Your AIX user identifier must belong to the AIX *cics* group to move both files. Renaming the uxa1 program enables you to distinguish it from the name used in a different database.

Now you must define the CICS/6000 resources by using SMIT or writing a shell script that adds all of the entries you need as shown in B.5.4, “Configure Programs and Transactions for CICS/6000 Using Script for Sybase System 10” on page 144.

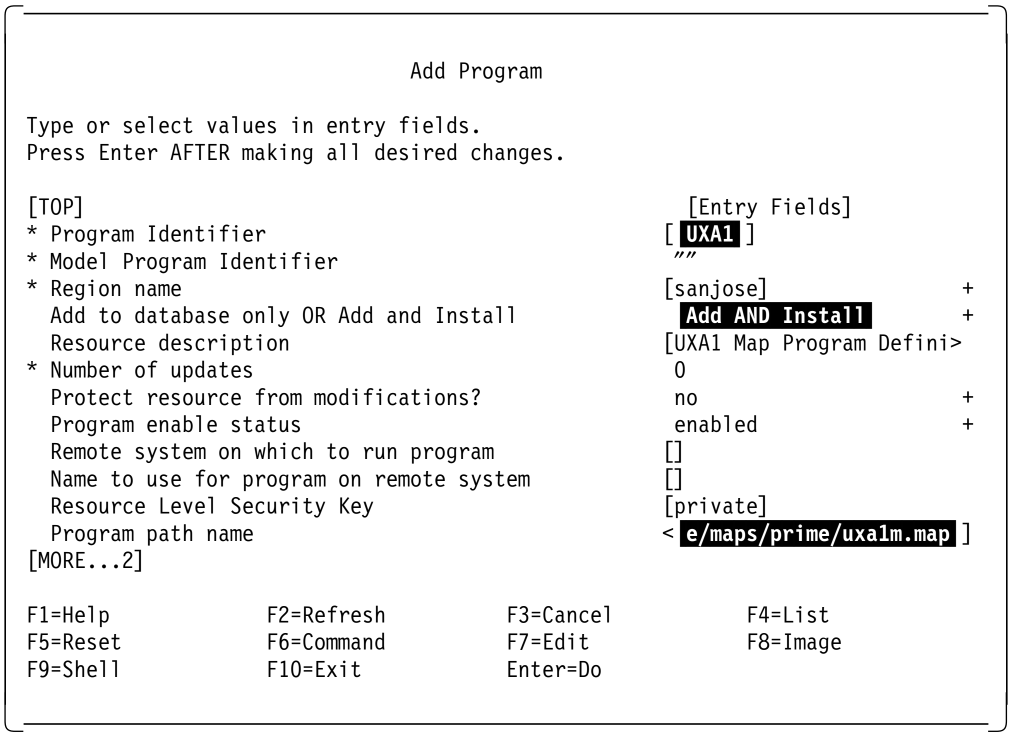
Resource definitions are held in the stanza files, which make up the permanent database that CICS/6000 uses to cold start the system, and in the run-time database, which CICS/6000 uses to hold information about its resources when it is running.

Enter the following command to define a program to the CICS/6000 stanza database:

```
$ smitty cics
   └→ Manage CICS/6000 Regions
      └→ Define Resources for a CICS/6000 Region
         └→ Manage Resource(s)
            └→ Programs
               └→ Add New
                  └→ Model Program Identifier
```

or use the fastpath command, *smitty cicsaddpd*.

When prompted for a model program identifier, choose the model "". You will get a panel similar to that shown in Figure 66 on page 142.



```
                              Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                    [Entry Fields]
* Program Identifier                                   [ UXA1 ]
* Model Program Identifier                              ""
* Region name                                          [sanjose]         +
  Add to database only OR Add and Install               Add AND Install  +
  Resource description                                 [UXA1 Map Program Defini>
* Number of updates                                     0
  Protect resource from modifications?                  no               +
  Program enable status                                 enabled          +
  Remote system on which to run program                []
  Name to use for program on remote system             []
  Resource Level Security Key                          [private]
  Program path name                                    < e/maps/prime/uxa1m.map ]
[MORE...2]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

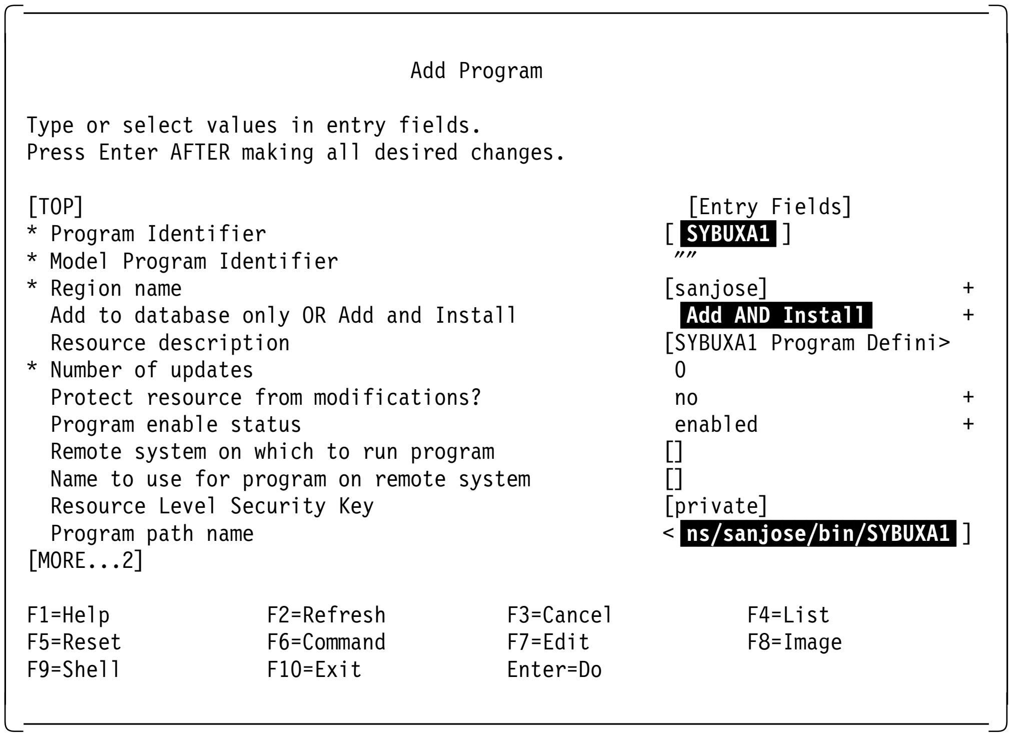
*Figure 66. Defining the UXA1 Map Program to CICS/6000: Sybase System 10*

The full program path name as shown in Figure 66 is: */var/cics_regions/<region_name>/maps/prime/uxa1m.map*.

**Note**

You must call the UXA1 program identifier; otherwise CICS/6000 returns an APCT error because the application defines a mapset UXA1.

You now have to install the *SYBUXA1* program by running the fastpath command, *smitty cicsaddpd*. You will get the panel shown in Figure 67 on page 143.


```
                                Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Program Identifier                                 [ SYBUXA1 ]
* Model Program Identifier                             ""
* Region name                                        [sanjose]          +
  Add to database only OR Add and Install              Add AND Install  +
  Resource description                               [SYBUXA1 Program Defini>
* Number of updates                                    0
  Protect resource from modifications?                 no               +
  Program enable status                                enabled          +
  Remote system on which to run program              []
  Name to use for program on remote system           []
  Resource Level Security Key                        [private]
  Program path name                                  < ns/sanjose/bin/SYBUXA1 ]
[MORE...2]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 67. Defining the SYBUXA1 Program to CICS/6000*

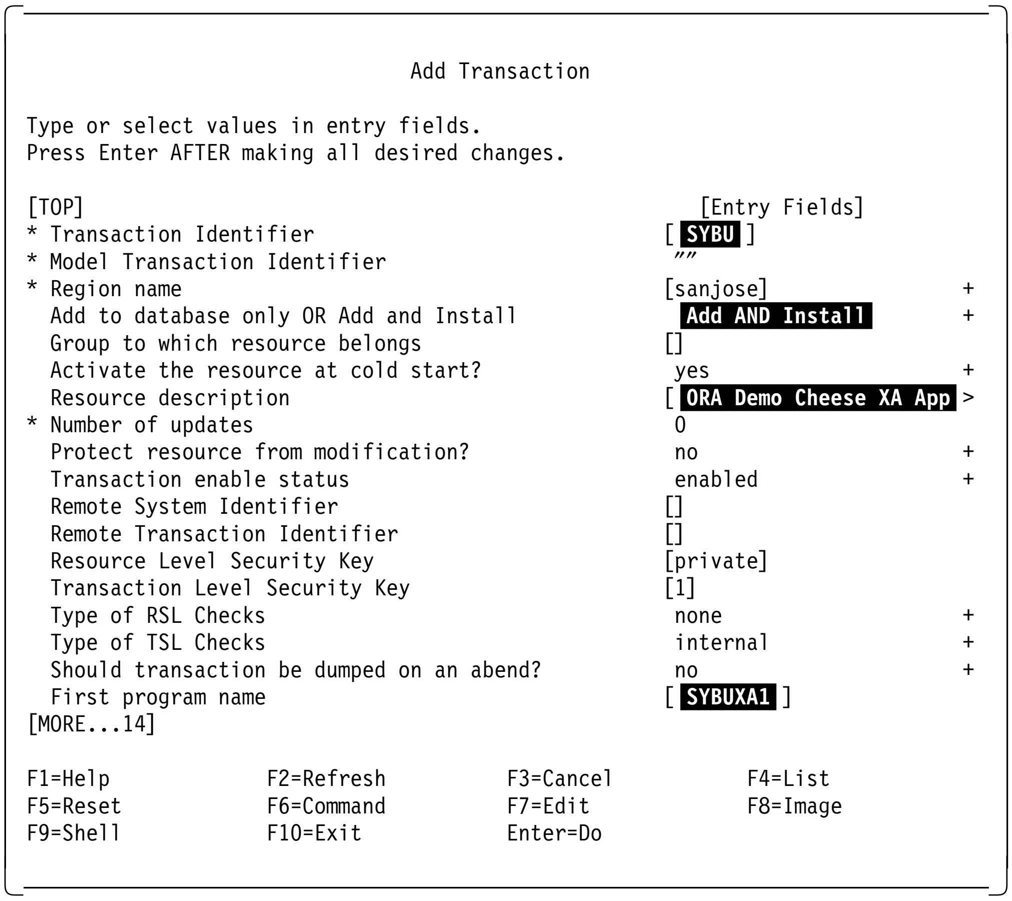
The full program path name as shown in Figure 45 on page 122 is: */var/cics_regions/<region_name>/bin/SYBUXA1*.

At this point you have to define the *SYBU* transaction to CICS/6000 by entering the following command:

```
$ smitty cics
   └─► Manage CICS/6000 Regions
      └─► Define Resources for a CICS/6000 Region
         └─► Manage Resource(s)
            └─► Transactions
               └─► Add New
                  └─► Model Transaction identifier
```

or by using the fastpath command: *smitty cicsaddtd*.

When prompted for a model transaction identifier, choose the model **""**. You will get a panel similar to that shown in Figure 68 on page 144.



```
                                Add Transaction

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                       [Entry Fields]
* Transaction Identifier                                [ SYBU ]
* Model Transaction Identifier                           ""
* Region name                                           [sanjose]             +
  Add to database only OR Add and Install                 Add AND Install     +
  Group to which resource belongs                       []
  Activate the resource at cold start?                   yes                  +
  Resource description                                  [ ORA Demo Cheese XA App >
* Number of updates                                      0
  Protect resource from modification?                    no                   +
  Transaction enable status                              enabled              +
  Remote System Identifier                              []
  Remote Transaction Identifier                         []
  Resource Level Security Key                           [private]
  Transaction Level Security Key                        [1]
  Type of RSL Checks                                     none                 +
  Type of TSL Checks                                     internal             +
  Should transaction be dumped on an abend?              no                   +
  First program name                                    [ SYBUXA1 ]
[MORE...14]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

*Figure 68. Defining the SYBU Transaction to CICS/6000*

## B.5.4 Configure Programs and Transactions for CICS/6000 Using Script for Sybase System 10

You can configure programs and transaction by writing a shell script as shown in Figure 69 on page 145.

```
#!/bin/ksh
echo $CICSREGION
BINDIR=/var/cics_regions/$CICSREGION/bin
MAPDIR=/var/cics_regions/$CICSREGION/maps/prime

cp uxa1 $BINDIR/SYBUXA1
cp uxa1m.map $MAPDIR

cicsadd -c td -r $CICSREGION \
                ResourceDescription="SYB Demo Cheese XA Application" \
                ProgName="SYBUXA1" \
                SYBU

cicsadd -c pd -r $CICSREGION -P "SYBUXA1" \
                ResourceDescription="SYBUXA1 Program Definition" \
                PathName="$BINDIR/SYBUXA1"

cicsadd -c pd -r $CICSREGION -P "UXA1" \
                ResourceDescription="UXA1 Map Definitions" \
                PathName="$MAPDIR/uxa1m.map" \
                ProgType=map
```

*Figure 69. Sample Shell Script to Configure CICS/6000 Resources: Sybase System 10*

These configurations take effect only after a cold start of CICS/6000 because we have defined resources only to database stanza files.

# Glossary

## A

**API**. Application programming interface. A set of calling conventions defining how a service is invoked through a software package.

**APPC**. Advanced Program-to-Program Communication. An implementation of SNA′s LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

**archive**. The maintenance and use of backup copies of files or full disk images to be used in the case of loss or damage of data due to unforeseen circumstances. Archived copies are also used to maintain historical copies of information.

**asynchronous**. Without regular time relationship; unexpected or unpredictable with respect to the execution of program instruction. See synchronous.

## B

**business process**. An entity-handling activity that is of limited duration, defined in scope, and set by business goals and policies, not by organization or implementation.

## C

**client**. As in client-server computing, the application that makes requests to the server and, often, deals with the interaction necessary with the user.

**client-server computing**. A form of distributed processing, in which the task required to be processed is accomplished by a client portion that requests services and a server portion that fullfills those requests. The client and server remain transparent to each other in terms of location and platform. See distributed processing, client, and server.

**communications models**. The ways of implementing communications between the distributed elements of an application. Examples are conversational, remote procedure call, and messaging.

**conversational**. Communication model where two distributed applications exchange information by way of a conversation; typically one application starts (or allocates) the conversation, sends some data, and allows the other application to send some data. Both applications continue in turn until one decides to finish (or deallocate). The conversational model is a synchronous form of communication.

**cooperative processing**. The process by which a single application is divided between two or more hardware platforms. Very often the term is used to reflect a tightly coupled relationship between the parts of the application.

**CPI-C**. Common Programming Interface for Communication, providing a common interface for the implementation of the conversational communication model.

## D

**database**. (1) A collection of interrelated data stored together with controlled redundancy according to a scheme to serve one or more applications. (2) All data files stored in the system. (3) A set of data stored together and managed by a database management system.

**DCE**. Distributed Computing Environment. Adopted by the industry as a de facto standard for distributed computing. DCE allows computers from a variety of vendors to communicate transparently and share such resources as computing power, files, printers, and other objects in the network.

**design**. The process of composing a software blueprint, showing how to build from the requirements specification document. Design often includes module decompositions, data structure definitions, file format definitions, and important algorithm descriptions.

**directory**. A repository of information about objects, functions, or services and a set of services to manipulate the repository. In this document the term directory is used to describe a function to identify and provide information about resources in a network, such as servers, files, applications.

**distributed processing**. Distributed processing is an application and/or systems model in which function and data may be distributed across multiple computing resources connected on a LAN or WAN. See client-server computing and peer-to-peer processing.

## E

**ENCINA**. Enterprise computing in a new age. A set of DCE-based products from Transarc Corporation that are available on the RISC System/6000. Encina is a family of online transaction processing products and includes:

- Encina Toolkit Executive
- Encina Server

- Encina Structured File Server (SFS)
- Encina Peer-to-Peer Communication Executive (PPC).

**environment**. The collective hardware and software configuration of a system.

**event driven**. Term used to describe an application environment where the sequence of processing is determined primarily by the application′s response to external events, which may be initiated by the user, the operating system, or other applications.

## F

**file server**. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

## G

**GUI**. Graphical user interface. A style of user interface that replaces the character-based screen with an all-points-addressable, high resolution graphics screen that uses windows to display multiple applications at the same time while allowing user input by means of a keyboard or a pointing device such as a mouse, a pen, or a trackball.

## H

**heterogeneous network**. A local or wide area network comprising hardware and software from different vendors, usually implementing multiple protocols, operating systems, and applications.

**hook**. (1) In programming, an area of program code that makes connections with other program codes. (2) A mechanism by which procedures are called when certain events occur in the system. (3) A documented programming interface in a component of system or application software that provides a means of extending or modifying the function of that component through the attachment of user-supplied code.

**host**. (1) In a computer network, a computer providing services such as computation, database access, and network control functions. (2) The primary or controlling computer in a multiple computer installation.

## I

**instance**. An individual application object that belongs to a particular object class. An instance of a class is an actual object displaying the properties of that class.

**instantiate**. To represent a class abstraction with a concrete instance of the class by providing values for the attributes of the class.

**interactive processing**. A type of processing in which a program or system alternately accepts input and responds. An interactive system is conversational, that is, a continuous dialog exists between the user and the system.

**interoperability**. The ability to interconnect systems from different manufacturers and have them work together to satisfy a business requirement. Some examples of requirements are message interchange between systems, and sharing of resources, such as data, between applications running on different hardware and software platforms.

## L

**LUW**. Logical unit of work. An update that durably transforms a resource from one consistent state to another consistent state.

## M

**messaging**. A communications model whereby the distributed applications communicate by sending messages to each other. A message is typically a short packet of information that does not necessarily require a reply. Messaging implements asynchronous communications.

**middleware**. Middleware is a set of services that allows distributed applications to interoperate on a LAN or WAN. It shields the developer or end-user from the system complexity and enables delivery of service requests or responses transparently across computing resources.

## O

**object**. A program or a group of data that can behave like a thing in the real world.

**OLTP**. Online transaction processing. A style of computing that supports interactive applications in which requests submitted by terminal users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A online transaction processing system supervises the sharing of resources for processing multiple transactions at the same time, minimizes compute time and duration of locks and, separates user thinking-time from the use of storage and other resources.

## P

**peer-to-peer processing**. A form of distributed processing where the distributed parts of an application can both make and serve requests, typically in a conversational structure, as opposed to one side making the requests and the other side filling them, as in client-server computing.

**portability**. The ability to move application software components from one system for use on another system. Perfect portability would permit such movement without modification of those components.

**process**. (1) A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. (2) Any operation or combination of operations on data. (3) A function being performed or waiting to be performed. (4) A program in operation, for example, a daemon.

**protocol**. (1) A formal set of conventions governing the format and control of data. (2) A set of procedures or rules for establishing and controlling transmissions from a source device or process to a target device or process.

## R

**recovery**. The use of archived copies to reconstruct files, databases, or complete disk images after they are lost or destroyed.

**RPC**. Remote procedure call. A communication model where requests are made by function calls to distributed procedures elsewhere. The location of the procedures is transparent to the calling application.

**resource manager**. A software program that maintains the state of resources and provides access and control to them through APIs. A resource can be a device as well as a program or object, although normally it is referred to as a device.

## S

**server**. Any computing resource dedicated to responding to client requests. Servers may be linked to clients through LANs or WANs to perform services, such as printing, database access, fax, and image processing, on behalf of multiple clients at the same time.

**SQL**. Structured query language. SQL started as IBM′s query language for DB2. SQL became so popular with users and vendors outside IBM that ANSI adopted a version of SQL as a U.S. standard in 1986. A year later ISC gave SQL formal international standard status.

**stored procedures**. Facility for storing procedural code associated with relational database management systems (RDBMSs) that enforces its use during any database operation.

**synchronous**. (1) Pertaining to two or more processes that depend on the occurrences of a specific event such as a common timing signal. (2) Occurring with a regular or predictable time relationship.

## T

**test**. Testing involves checking each individual module built during the implementation phase, then integrating the modules into a single program structure. The program as a whole is then tested to ensure that it performs as designed.

**thread of control**. Inside the X/Open DTP model, the concept that associates RM work with the global transaction. Routines in the XA interface that manage the association among a thread of control and transactions must be called from the same thread.

**transaction**. A unit of processing (consisting of one or more application programs) initiated by a single request. A transaction may require the initiation of one or more tasks for its execution.

**transaction branch**. A part of the work in support of a global transaction for which the TM and the RM engage in a commitment protocol coordinated with, but separate from, that for other branches.

**transaction manager**. Provides the function to begin, end, commit, and rollback transactions.

**transaction monitor**. Provides a total environment for transactional applications. In addition to transaction manager functions, provides services to aid development, execution, and operation of transaction applications.

**transaction processing**. A style of computing that supports interactive applications in which request submitted by users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A transaction processing system supervises the sharing of resources for processing multiple transactions at the same time.

**transparency**. The state where certain elements of the system are hidden so that other parts of the system need not depend on the specific implementation of those elements. For instance, a servers′s location is transparent to its client; thus, when the server′s location changes, the clients need not be changed.

**two-phase commit**. For a database, a protocol that is used to ensure uniform transaction commit or abort in a distributed data environment between two or more participants. The protocol consists of two phases: the first to reach a common decision, and the second to implement the decision.

**TX**. Within the DTP model adopted by X/Open, the interface among the application or transaction monitor and the transaction manager.

## W

**workflow**. The automation of work among users where the system is intelligent enough to act based on the definition of work types, tasks, and the recognition of dynamic processing conditions.

**workstation**. A configuration of input/output equipment at which an operator works. A terminal or microcomputer, usually one that is connected to a mainframe or a network, at which a user can perform applications.

## X

**XA**. Within the DTP model adopted by X/Open, the interface between the transaction manager and resource managers.

# List of Abbreviations

| | |
|---|---|
| ***API*** | application programming interface |
| ***ATI*** | automatic task initiation |
| ***CRA*** | crash recovery archive |
| ***EI*** | EXEC interface |
| ***FD*** | file definition |
| ***IBM*** | International Business Machines Corporation |
| ***ISC*** | intersystem communication |
| ***ITSO*** | International Technical Support Organization |
| ***LAN*** | local area network |
| ***LPP*** | licenced program product |
| ***LUW*** | logical unit of work |
| ***OLTP*** | online transaction processing |
| ***PPC*** | peer-to-peer communication |
| ***PID*** | process identifier |
| ***RD*** | region definition |
| ***RDBMS*** | relational database management system |
| ***SFS*** | Structured File Server |
| ***SMIT*** | System Management Interface Tool |
| ***SRC*** | system resource controller |
| ***transid*** | transaction identification code |
| ***UD*** | user definition |
| ***WAN*** | wide area network |

# Index

## ITSO Technical Bulletin Evaluation RED000

**AIX CICS/6000 and**
**Relational Database Management Systems Integration:**
**Experiences with the XA Interface**

**Publication No. GG24-4214-00**

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

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

**Please rate on a scale of 1 to 5 the subjects below.**
**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

**Overall Satisfaction** ____

| | | | |
|---|---|---|---|
| Organization of the book | ____ | Grammar/punctuation/spelling | ____ |
| Accuracy of the information | ____ | Ease of reading and understanding | ____ |
| Relevance of the information | ____ | Ease of finding information | ____ |
| Completeness of the information | ____ | Level of technical detail | ____ |
| Value of illustrations | ____ | Print quality | ____ |

**Please answer the following questions:**

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

______________________________

______________________________

What other topics would you like to see in this Bulletin?

______________________________

______________________________

What other Technical Bulletins would you like to see published?

______________________________

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

______________________________ ______________________________
Name Address

______________________________ ______________________________
Company or Organization

______________________________ ______________________________
Phone No.

**ITSO Technical Bulletin Evaluation** **RED000**
GG24-4214-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 471, Building 070B
5600 COTTLE ROAD
SAN JOSE CA
USA 95193-0001

Fold and Tape **Please do not staple** Fold and Tape

GG24-4214-00

Cut or Fold
Along Line

Printed in U.S.A.

GG24-4214-00