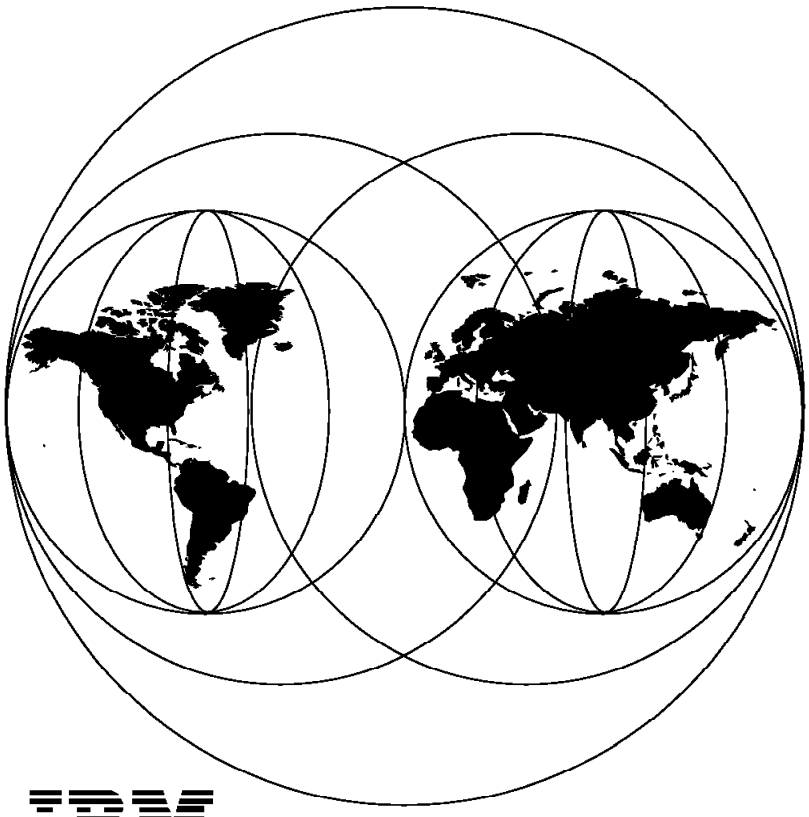
International Technical Support Organization

SG24-4558-00

DB2 for MVS Connections with AIX and OS/2

November 1995



IBM

International Technical Support Organization

San Jose Center

International Technical Support Organization

SG24-4558-00

DB2 for MVS Connections with AIX and OS/2

November 1995

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 (November 1995)

This edition applies to Version 2.1 of DB2 common server, version 2.3 of DDCS and version 3.1 of DB2 for MVS/ESA for use with the OS/2, AIX and MVS/ESA operating systems respectively.

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 80-E2
650 Harry Road
San Jose, California 95120

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 is the result of a residency project conducted at the IBM International Technical Support Organization - San Jose Center to investigate the new features of DB2 common server V2.1 and DDCS V2.3. It focuses primarily on what is new with these products but also covers DRDA concepts in general when operating in the AIX, OS/2, and MVS DB2 platforms.

This document is written for IBM technical professionals and customer technical personnel responsible for establishing DRDA connections among AIX, OS/2, and MVS. Knowledge of DRDA principles and familiarity with the OS/2, AIX, and MVS operating environments are assumed but not necessary.

(133 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help database administrators and application programmers establish DRDA connections among OS/2, AIX, and MVS. The information in this publication is not intended as the specification of any platforms programming interfaces that are provided by DRDA or the products supporting it. See the PUBLICATIONS section of the IBM Programming Announcement for the following products for more information on the formal product documentation:

- DATABASE 2 for MVS/ESA
- DATABASE 2 for OS/2
- DATABASE 2 for AIX
- Distributed Database Connection Services for OS/2
- Distributed Database Connection Services for AIX
- Communications Manager/2
- SNA Server/6000
- ACF/VTAM
- RACF

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

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

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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

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

ACF/VTAM
APPN
CICS
DB2 for OS/2
DB2 for AIX
IBM
MVS/ESA
RACF
SQL/DS
AIX
AS/400
DB2 for MVS/ESA
DB2 for OS/400
DRDA
IMS
OS/2
RS/6000
VTAM

The following terms are trademarks of other companies:

ENCINA is a trademark of Transarc Corporation.
Windows is a trademark of Microsoft Corporation.

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

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

C-bus is a trademark of Corollary, Inc.

Other trademarks are trademarks of their respective companies.

Preface

This document will help customers understand the new features of DB2 common server V2.1 and DDCS V2.3 and establish DRDA connections among the OS/2, AIX, and MVS platforms.

It describes various DRDA connectivity scenarios, explains how to implement them, and reviews various issues that may arise as a result of implementation.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, “Introduction”

  In this chapter we provide an overview of the Distributed Relational Database Architecture (DRDA) and the IBM products that implement it. We also describe the various DRDA scenarios, and the prerequisite products to implement those scenarios. Lastly, in this chapter we provide a brief overview of the various new features and functions that DB2 common server V2.1 and DDCS V2.3 provide.

- Chapter 2, “Connectivity”

  In this chapter we discuss several DRDA configurations and how to implement them in a LAN environment. We cover how to configure the various DB2 and DDCS products to establish connections from the workstation to the host and from the host to the workstation.

- Chapter 3, “Binding”

  In this chapter we describe the difference between binding on the workstation and the host. We also give a brief overview of how to bind some handy utilities.

- Chapter 4, “Security Considerations”

  In this chapter we explain the different locations where security can be implemented and how to combine the security settings to provide the desired level of security. We also cover some case-sensitivity issues to be considered when using DRDA connections.

- Chapter 5, “Distributed Unit of Work”

  In this chapter we describe distributed unit of work and how it is implemented on the various platforms. We cover the various combinations of DB2 common server and DB2 for MVS/ESA in a DUW. We also explain some new SQL statements for the DUW environment, as well as some new precompile options for DB2 common server.

- Chapter 6, “Stored Procedures”

  In this chapter we explain what a stored procedure is and how it can be invoked. We compare the implementation of the new SQL CALL statement with the existing DARI API and discuss the merits of both. Some sample stored procedures in DB2 for MVS/ESA are also briefly examined.

- Chapter 7, “New Functions in DDCS and DB2 common server”

  In this chapter we cover various concepts ranging from compound SQL, accounting, tracing, heap size, directory caching, and hopping.

Related Publications

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

- *DATABASE 2 Administration Guide, Volume I*, SC26-4888
- *DATABASE 2 Administration Guide, Volume II*, SC26-4888
- *DATABASE 2 Administration Guide, Volume III*, SC26-4888
- *DATABASE 2 Application Programming and SQL Guide*, SC26-4889
- *DATABASE 2 Version 4 Application Programming and SQL Guide*, SC26-3266
- *DATABASE 2 SQL Reference*, SC26-4890
- *DATABASE 2 Command and Utility Reference*, SC26-4891
- *DATABASE 2 SQL Reference for common servers*, S20H-4665
- *DATABASE 2 Command Reference for common servers*, S20H-4645
- *DATABASE 2 Administration Guide for common servers*, S20H-4580
- *DATABASE 2 Application Programming Guide for common servers*, S20H-4643
- *Distributed Database Connection Services User's Guide for common servers*, S20H-4793
- *DATABASE 2 Installing and Using AIX Clients*, S20H-4666
- *DATABASE 2 Installing and Using OS/2 Clients*, S20H-4782
- *DATABASE 2 API Reference*, S20H-4984
- *DATABASE 2 AIX/6000 Administration Guide*, SC09-1571

International Technical Support Organization Publications

- *Distributed Relational Database: DB2 Multisite Update*, GG24-4153
- *DB2 for MVS DRDA Server: Security Considerations*, GG24-2500
- *Distributed Relational Database: Using DDCS/6000 DRDA Support with DB2 and DB2/400*, GG24-4155

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

International Technical Support Organization Bibliography of Redbooks, SG24-3070.

To get a catalog of ITSO redbooks, VNET users may type:

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

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOPUB LISTALLX. This package is updated monthly.

How to Order ITSO Redbooks

IBM employees in the 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 contact their local IBM office. Guidance may be obtained by sending a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

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

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser (such as WebExplorer from the OS/2 3.0 Warp BonusPak) to the following:

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

IBM employees may access LIST3820s of redbooks as well. Point your web browser to the IBM Redbooks home page:

```
http://w3.itsc.pok.ibm.com/redbooks/redbooks.html
```

Acknowledgments

This project was designed and managed by:

Silvio Podcameni
International Technical Support Organization, San Jose Center

The authors of this document are:

Craig Edwards - ISSC Australia

Paulo Sergio Pereira Garcia - IBM Brazil

This publication is the result of a residency conducted at the International Technical Support Organization, San Jose Center.

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

Peter Shum - Toronto Laboratory
Corinne Baragoin - IBM France
Guillermo Kaminker - IBM Argentina
Frantz De Rycke - IBM Canada

Thanks also to Maggie Cutler for her editorial assistance.

Silvio Podcameni, Project Leader
ITSO - San Jose

Chapter 1. Introduction

In this chapter we discuss the following topics:

- Distributed Relational Database Architecture
- Product packaging
- What's new with DB2 common server?
- What's new with DDCS?

1.1 Distributed Relational Database Architecture

Distributed Relational Database Architecture (DRDA) is an open architecture that describes a database protocol to exchange data between different remote database systems. Through DRDA an application in one system can query and update data in another system, in a consistent manner, with functions to provide security and integrity of databases.

DRDA was designed to be integrated with existing network and system facilities. It is based on the following IBM architectures:

- SNA Management Services Architecture (MSA)
- SNA Logical Unit type 6.2 protocol (LU 6.2)
- Distributed Data Management (DDM)
- Formatted Data Object Content Architecture (FD-OCA)
- Character Data Representation Architecture (CDRA)

MSA Transfers and manages alerts. Defines a consistent method of sending and recording error messages.

LU 6.2 Also known as Advanced Program-to-Program Communication (APPC). Provides communication functions such as authentication, error alerts, and session security.

DDM Describes the model of the database environment. Provides commands, parameters, data objects, and reply messages that work as an intermediate language that all systems understand.

FD-OCA Provides the description of the data with information about data types and representation.

CDRA Provides character data integrity when data is transmitted between systems with different encoding.

Many IBM and non-IBM products implement DRDA today. IBM has relational database manager products that run in different platforms. With DRDA it is possible to use those products together and take advantage of their unique capabilities.

The IBM relational database products that support DRDA are:

- IBM DATABASE 2 MVS (DB2 for MVS/ESA)
- IBM DATABASE 2 VM (DB2 for VM)
- IBM DATABASE 2 VSE (DB2 for VSE)

- IBM DATABASE 2 OS/400 (DB2 for OS/400)
- IBM DATABASE 2 AIX (DB2 for AIX)
- IBM DATABASE 2 OS/2 (DB2 for OS/2)
- IBM DATABASE 2 for the Solaris Operating System
- IBM DATABASE 2 for HP-UX (DB2 for HP-UX).

The Distributed Database Connection Services (DDCS) products provide DRDA application requester (AR) functions in the workstation environment. These functions allow the DB2 common server products to connect to DRDA application servers (ASs), such as DB2 for MVS/ESA, DB2 for VSE and VM, and DB2 for OS/400 (see Figure 1).

DB2 common server is the collective name by which the DB2 products for the UNIX and Intel-based architectures are known and includes the following operating systems: DOS, Windows, OS/2, AIX, Solaris, and HP-UX.

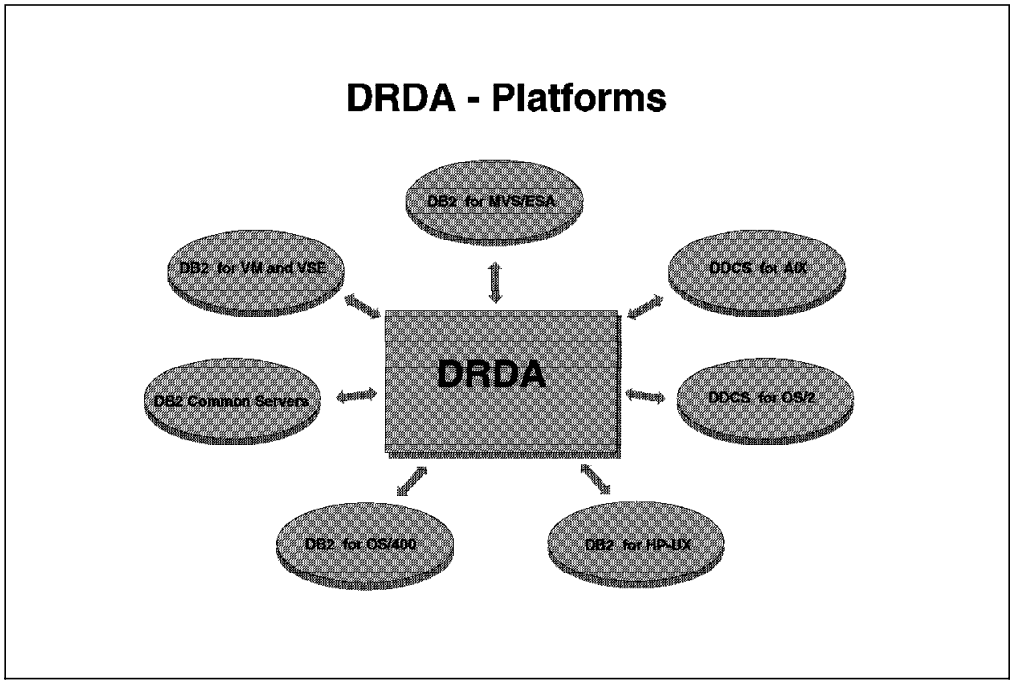


Figure 1. DRDA Platforms

DDCS implements the AR functions as specified by DRDA. It provides remote unit of work function to applications running from DB2 common server client workstations. In DDCS V2.3, distributed unit of work (DUW) with multisite read or single-site update and multisite read is also supported. This is the first step towards fully supporting DRDA level 2. DRDA level 2 DUW is supported through DDCS if a transaction manager such as ENCINA is used in conjunction with DDCS.

1.2 Product Packaging

In this section, we look at the prerequisite DB2 and DDCS product combinations for several DRDA environments (see Table 1). We describe the environments and the services provided for each environment.

Table 1. DB2 and DDCS Product Combinations for DRDA Environment

Environment	Capabilities	Product
DRDA AS on workstation	Workstation accepts inbound connections from DRDA ARs such as DB2 for MVS/ESA and DB2 for OS/400. See Figure 2 on page 4.	DB2 common server
DRDA AR on workstation	Workstation issues DRDA requests to DRDA ASs such as DB2 for MVS/ESA and DB2 for OS/400. See Figure 3 on page 4.	DDCS single-user or DDCS multi-user
DRDA gateway from workstation to host	Workstation propagates incoming requests from remote clients to a DRDA AS, such as DB2 for MVS/ESA and DB2 for OS/400. See Figure 4 on page 5.	DDCS multi-user
DB2 client to gateway	Workstation allows a client to connect to another DB2 system. The connection in the LAN is made using the DB2 private protocol, *not* the DRDA protocol. An example would be an OS/2 client connecting to a DB2 common server, which then passes the request to a DRDA AS. See Figure 4 on page 5.	• Client Application Enabler **or** • Software Development Kit **or** • DB2 common server **or** • DDCS multi-user
DRDA AS on MVS	Host accepts inbound connection from DRDA ARs, such as DB2 for OS/2, DB2 for AIX, and DB2 for OS/400. See Figure 4 on page 5.	DB2 for MVS/ESA
DRDA AR on MVS	Host issues DRDA requests to DRDA ASs such as DB2 for OS/2, DB2 for AIX, and DB2 for OS/400. See Figure 2 on page 4.	DB2 for MVS/ESA

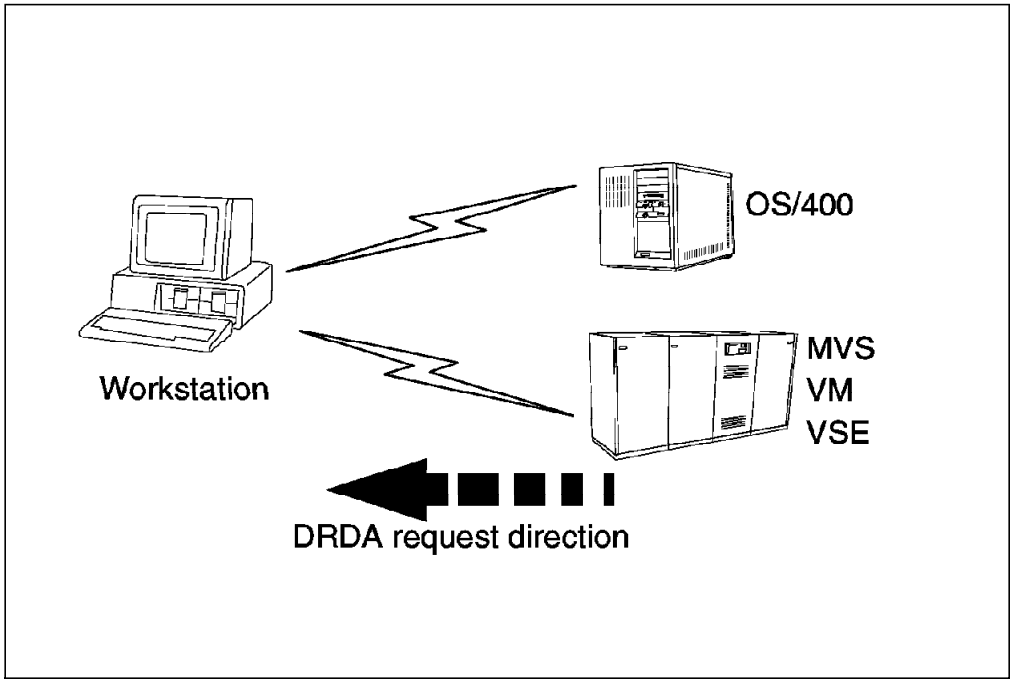


Figure 2. Host-to-Workstation Connection

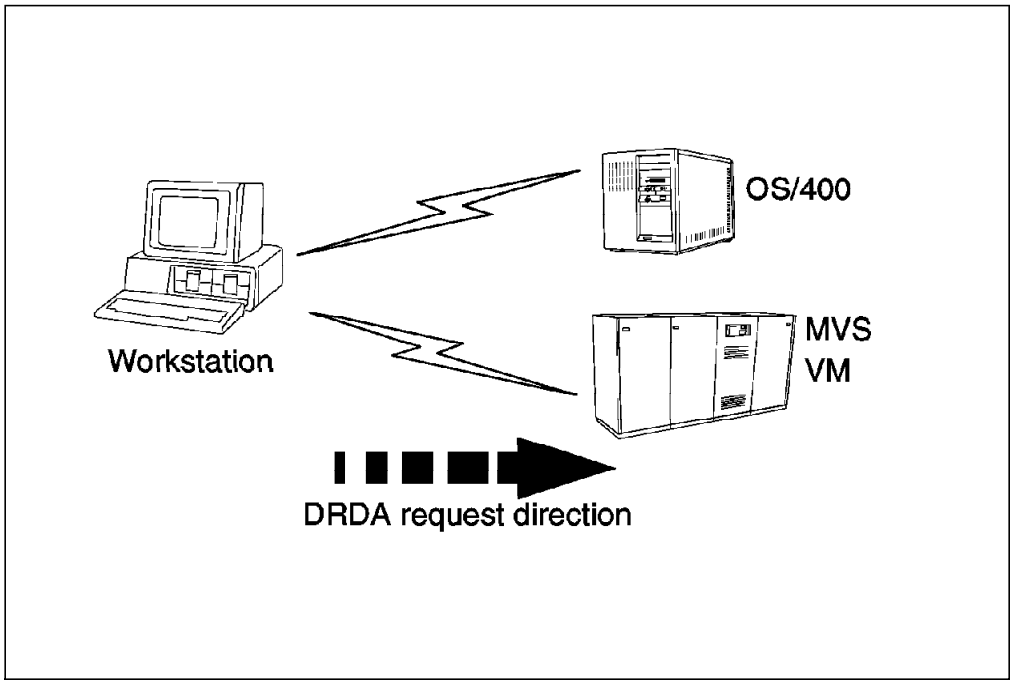


Figure 3. Workstation-to-Host Connection

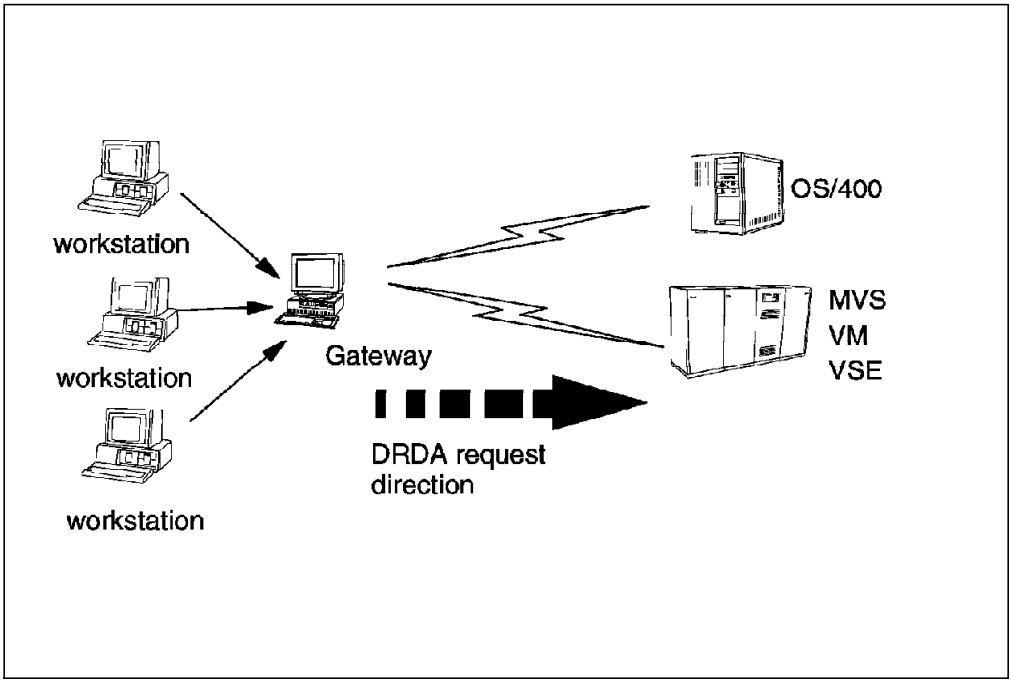


Figure 4. Workstation-to-Gateway-to-Host Connection

1.3 What′s New with DB2 common server?

The following enhancements to the DB2 common server products in V2.1 have specific relevance to DRDA:

- DRDA support

  The DB2 DRDA AS features provide support for DRDA level 1 and all required functions of DRDA level 2. With the DB2 DRDA AS feature installed, applications using a DRDA AR can create packages at the DB2 common server and subsequently execute those packages subject to SQL supported by the DB2 common server.

- Stored procedure support

  The objective of DRDA stored procedures is to support the ANSI standard SQL interface. Support for the SQL CALL statement improves the performance of applications that use DRDA access and provides increased integrity for the processing of data performed at the DRDA server.

- Directory caching

  This new feature improves connection performance by caching the entries for the system database directory, distributed connection service directory, and node directory. Subsequent use of the directory cache eliminates further I/O operations against the directory files and provides a significant improvement in connection elapsed time.

- SQLCODE remapping

  This new feature also improves connection performance. It reduces the overhead incurred by SQLCODE remapping at the DRDA AR. DRDA uses SQLCODE remapping to reconcile inconsistencies in the error handling characteristics of the DRDA AS. This enhancement allows SQLCODE remapping to be disabled where appropriate.

1.4 What′s New with DDCS?

The following enhancements have been added to DDCS V2.3.

- Removal of DB2 for OS/2 as a prerequisite

  DDCS for OS/2 no longer has the requirement that DB2 for OS/2 be licensed. Thus DDCS for both the AIX and OS/2 platforms can operate in a stand-alone capacity.

- DRDA bind options

  DDCS now supports more options on the PREP and BIND commands. Users of DDCS can exploit the binding options of database servers when using DRDA protocols to communicate with those servers.

- Compound SQL

  Support for compound SQL allows the application to group multiple SQL statements into a single executable block. These statements can then be sent for processing in a compact, continuous stream, thus reducing the network flow and response time.

- Accounting string support

  Accounting string support enables an accounting system to track DRDA AS resource usage through DDCS. An accounting identifier is sent to the DRDA

AS with each application's connect request. Because DDCS can forward accounting data to a DRDA AS, system administrators can associate resource usage with each user access and charge users accordingly.

Chapter 2. Connectivity

In this chapter we discuss how to connect MVS, AIX, and OS/2 platforms using the DRDA protocol, through a token-ring LAN environment.

2.1 SNA Concepts

Before you can understand the DRDA connections that are described in this book, you must familiarize yourself with the following SNA concepts:

- Logical unit
- Session
- Conversation
- Bind
- Mode name
- Transaction program name
- Side information
- Requesters and servers
- Data transmission process
- Network connection overview

We describe these concepts below.

2.1.1 Logical Unit

A logical unit (LU) represents a “user” of the network. It can be either software, hardware, or a combination of both. An LU is responsible for sending and receiving network messages on behalf of application programs and end users.

SNA defines many LU types. Some represent hardware devices, such as terminals, printers, or automatic teller machines (ATMs), and others represent network application programs. LU 6.2 is a type of LU that supports communication between network application programs in a distributed processing environment.

Some of the capabilities of an LU are dictated by the type of physical unit (PU) that controls the LU. For example, an LU under PU 2.0 is called a dependent LU, because in a VM, MVS, or VSE environment it depends on VTAM to communicate with any other network device.

An LU under PU 2.1 is called an independent LU because it can communicate with other PU 2.1 network devices without VTAM assistance (in a VM, MVS, or VSE environment) and with many partners (each could be a different database server or client) concurrently. For each partner an independent LU can have several sessions enabling multiple concurrent requests.
VTAM V3.4 or later supports dynamic independent LU creation.

Many network participants, including S/38, AS/400, and mainframes, are capable of supporting PU 2.1 connections. In cases where a communications controller is used to connect network participants, PU 2.1 must be supported by the

controller as well. If the controller does not support PU 2.1, PU 2.0 must be used, even though the network participants support PU 2.1.

2.1.2 Session

A session is a logical connection between two network addresses. Sessions are serially reusable by conversations. A session can support many conversations during its existence, but only one conversation at a time.

2.1.3 Conversation

An LU 6.2 conversation is a logical connection between two transaction programs. It is defined by the beginning and end of a dialog between two partners bound by the APPC ALLOCATE and APPC DEALLOCATE verbs.

Figure 5 shows that many conversations can exist in one session, but only one conversation can be active in a session. Conversation 3 is queued until the active conversation 2 ends.

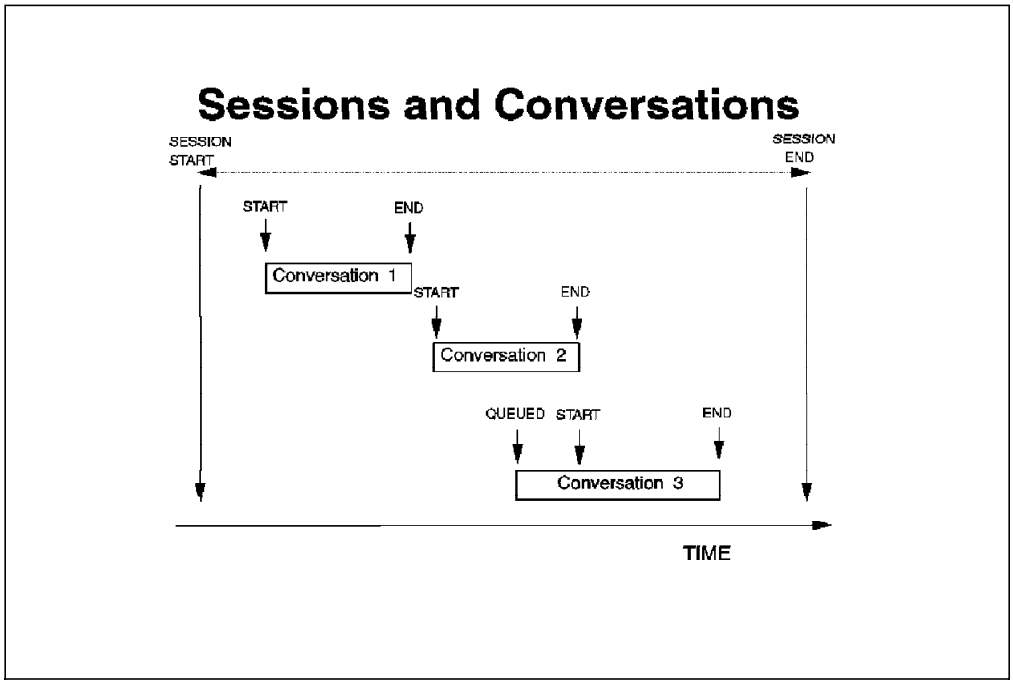


Figure 5. Sessions and Conversations

2.1.4 Bind

Bind is an SNA request unit that flows from the primary LU (the bind sender) to the secondary LU (the bind receiver). Bind is the first request unit to flow on an LU-LU session at session startup.

2.1.5 Mode Name

When establishing an LU-LU session, the communication system uses a mode that identifies the SNA session characteristics. The mode name is specified as an eight-character string. A given pair of LUs can establish sessions using several modes.

The mode parameters are:

- Request unit (RU) size

  Specifies the maximum length of data that one LU can to send to its session partner. See 2.1.9, “Data Transmission Process” on page 12.

- PACING

  Specifies that one LU sends a specified number of RUs and then waits for a “pacing response” from the receiving LU before it sends any more data. The number specified is the pacing window size (also called *pacing count*). See 2.1.9, “Data Transmission Process” on page 12.

- Class of service (COS)

  Defines the level of service the session receives and the route over which the session is established. A session for interactive users generally has a higher priority (and therefore a different COS) than an application doing large file transfers. The COS also defines a list of paths (virtual routes) through the network.

2.1.6 Transaction Program Name

The transaction program name (TPN) is part of the VTAM addressing scheme. It contains such information as where to find the executable for an application and its run-time characteristics. Each distributed database server is assigned a TPN. When an APPC application establishes a connection to another LU in the network, it must identify the TPN it wants to execute. The TPN identifies the program in the server system that receives the message. The DRDA TPN for DB2 for MVS/ESA is X′07F6C4C2′ (commonly referred to as X′07′6DB).

2.1.7 Side Information

The side information has information such as which is the partner LU and which should be the server application TPN. In OS/2, side information is known as CPIC side information, whereas in the AIX environment, it is known as LU 6.2 side information.

2.1.8 Requesters and Servers

Before a conversation between two applications can take place, a session between the two LUs that represent those applications must be established.

In the workstation environment (OS/2 and AIX), when an AR wants to communicate with another application, it uses a side information. This is the new way in which DDCS V2.3 implements the connection to a DRDA AS.

When the server system receives a request, it searches for a TPN that matches the TPN specified in the side information from the requester. Figure 6 on page 12 illustrates the relationship among these elements.

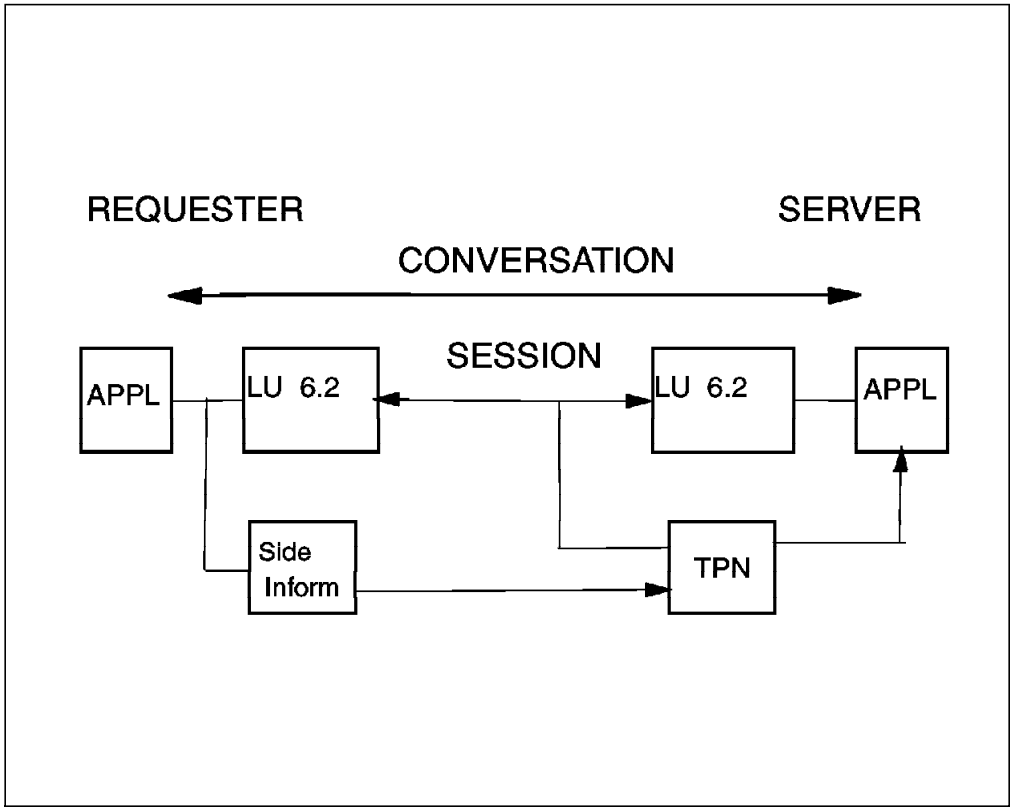


Figure 6. Requester and Server Environment

In a workstation environment the side information must be configured on the requester side, and the TPN must be configured on the server side. If a workstation is both a requester and a server, both the TPN and side information should be configured.

2.1.9 Data Transmission Process

The SNA data transmission process is designed to allow an SNA participant to transmit an arbitrary amount of data, without concern for the transmission capabilities of the devices in the network. If the data that the participant transmits exceeds the capacity of the devices, SNA divides the data into smaller units. SNA reassembles the data into its original format when it arrives at its final destination.

Figure 7 on page 13 shows that the data buffer to be transmitted can be divided into smaller units according to the amount of data to be transmitted and the RU size parameter. The RU size is specified during mode definition. SNA communications software adds a 3-byte prefix as the request header (RH) to the RU. The RH and RU make up a basic information unit (BIU). Before actually sending the BIU, the SNA communications software adds another prefix, the transmission header (TH), and transmits this information as a physical information unit (PIU).

If the data buffer is greater than the RU size, the SNA communications software may divide the data buffer into several PIUs before transmitting it.

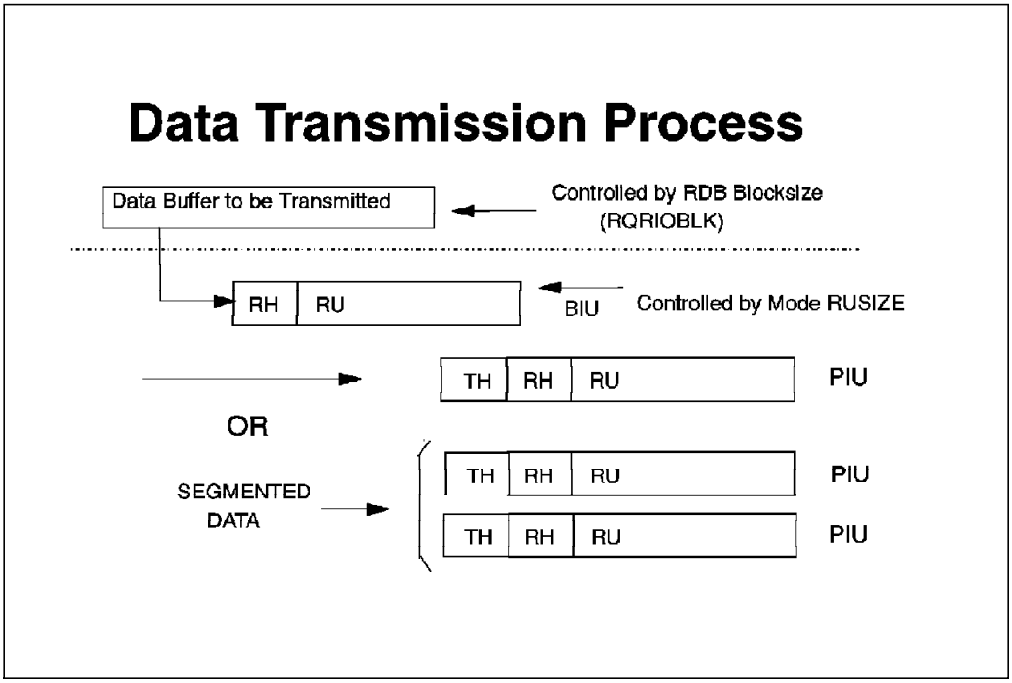


Figure 7. Units of Transmission

For a DDCS workstation, the size of the data buffer to be transmitted is defined in the RQRIOBLK parameter of the database manager configuration file. This parameter can be used to tune the DDCS workstation. The default DRDA block size is 32767 bytes. A large block size usually requires more memory on the DDCS workstation, thereby increasing the size of the working set, and may cause large amounts of paging on small workstations. However, a large block size can improve the performance of large requests. The block size does not usually affect the response time for small requests, such as a request for a single row of data.

Pacing allows the receiving application to control the rate at which the sending application transmits data. This control is accomplished by defining a pacing window (see 2.1.5, “Mode Name” on page 10). Failure to define pacing properly in a distributed database environment can cause severe network problems.

2.1.10 Network Connection Overview

In this section we review the communication requirements to connect from DB2 common server to DB2 for MVS/ESA. Figure 8 on page 14 shows a workstation connected to a token-ring LAN that is using a 37XX communications controller.

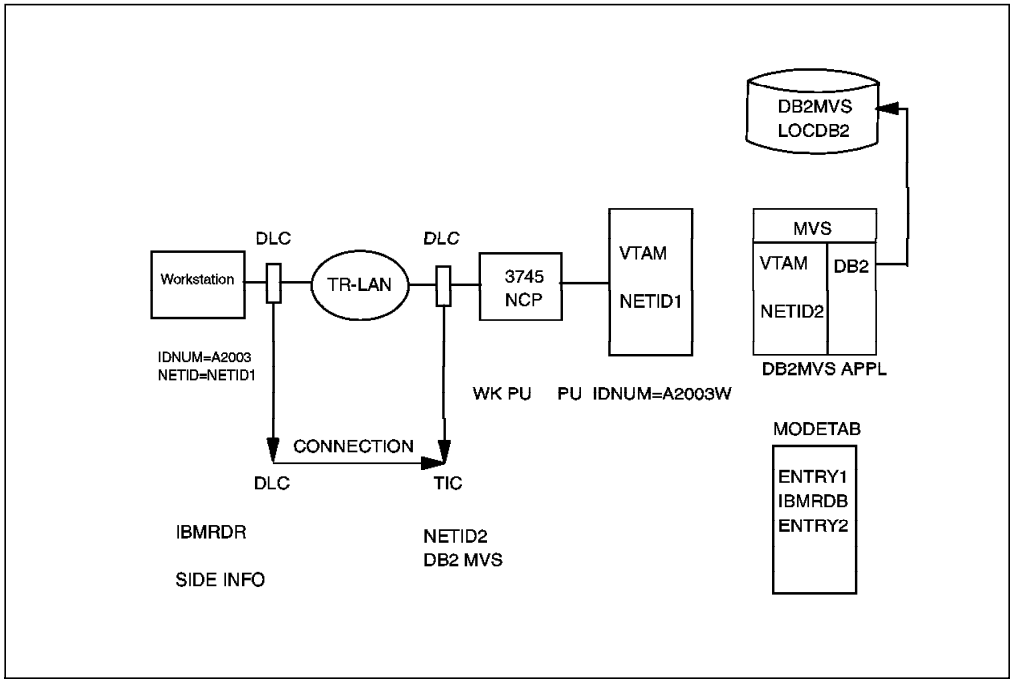


Figure 8. DB2 common server to DB2 for MVS/ESA: Network Overview

There must be some definitions in the adjacent VTAM (the VTAM where the workstation is defined and that is used to enable access from the DB2 common server to the network). You must define a VTAM PU to enable access from the workstation. There may be several PU definitions in VTAM. Each workstation maps to one definition when the DB2 common server establishes a connection to the adjacent VTAM.

The mapping is based on the IDNUM parameter of the PU macro. In the SNA communication products, for the workstation environment you specify the IDNUM when establishing communication with the adjacent VTAM. The NETID of the adjacent VTAM must also match the NETID specified in the SNA communication product. You must also state which data link control (DLC) the workstation uses to connect to the 37XX. The DLC could be an Ethernet card; for our example it is the TR DLC. The workstation might have more than one TR card, so you must state which card is used for the connection with the 37XX.

You must specify not only from where a message leaves the workstation, but also where the message arrives at the 37XX. So you must define a connection, where you specify the NETID of the adjacent VTAM as well as the TR address of the 37XX. This TR address is defined in the TIC parameter of the NCP.

So far the message can get to the network. But remember you must establish a session between two LUs: the LU that represents DB2 for MVS/ESA and the LU that represents the workstation.

In MVS, the LU for DB2 is specified as an APPL VTAM macro. When the communication component of DB2 for MVS/ESA comes up, it must identify itself to VTAM by using an LU name. The LU name that DB2 identifies to VTAM is registered in the boot strap data set (BSDS). On the workstation platform, the NETID and the LU name of DB2 for MVS/ESA must be specified as partners under the connection to the 37XX.

To establish a conversation, a mode that is available to DB2 for MVS/ESA must also be configured in the SNA communication product. The modes available to DB2 for MVS/ESA are kept as entries in a link-edited table pointed to by the MODETAB parameter of the APPL macro.

Finally, with DB2 common server, you must configure the side information; that is, the partner LU, mode name, and the TPN to establish the conversation. The TPN is X′07′6DB.

To summarize:

1. The physical link between the workstation and the host is established by using the DLC in the workstation and the token-ring interface connector (TIC) NCP definition for the host controller.
2. The session is an LU-to-LU relationship.
3. The two commmunication application programs exchange messages by using a conversation.

Note: If the workstation and DB2 for MVS/ESA are on different NETIDs, some cross-domain definitions are also required.

2.2 Configuring DDCS for OS/2 As a DRDA AR

In this section, we outline the steps to define and establish a DRDA connection by using DDCS for OS/2 from an OS/2 workstation to an MVS host. The steps are:

1. Configure Communications Manager/2 (CM/2)

   DDCS for OS/2 now uses CPIC side information to store the details for each remote partner. This change has some local LU implications.

2. Configure DDCS for OS/2

   The new release of DDCS for OS/2 has a completely new graphical user interface (GUI) to the directories. The new GUI, called the database director, provides the same function as the previous directory tool but has a different presentation interface.

3. Carry out post-setup activities

   After you have configured CM/2 and DDCS for OS/2, you must carry out some other tasks, such as ensuring that the DB2 for MVS/ESA system has been configured correctly and binding packages to the remote database.

2.2.1 Configuring CM/2

The DRDA protocol uses APPC to establish connections to remote DB2 servers. CM/2 provides the interface to APPC in the OS/2 environment. In order for CM/2 to manage your APPC environment, you must define several CM/2 profiles.

2.2.1.1 Defining a Local Node

First you must define the attributes of your workstation. These attributes may already have been defined for you when CM/2 was installed. If not, use the Local Node Characteristics window in the CM/2 setup (see Figure 9 on page 16).

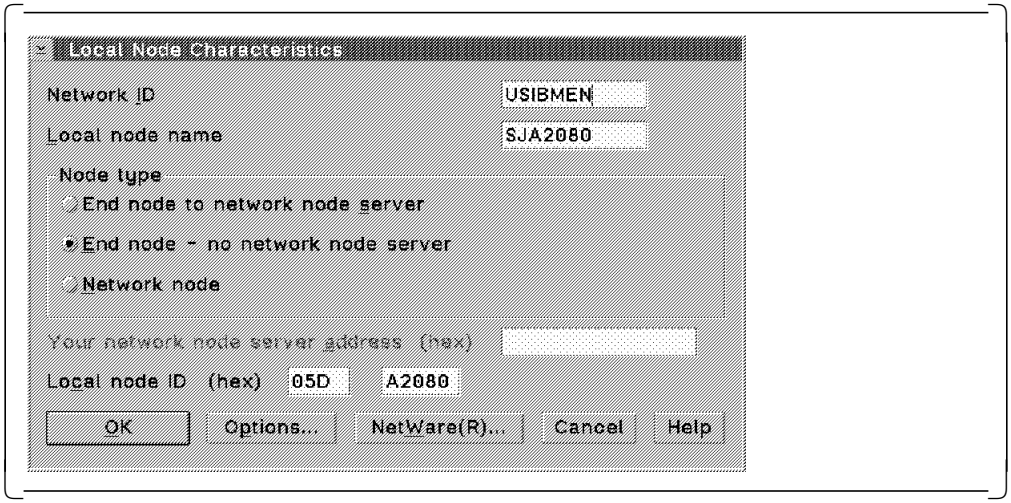


Figure 9. Local Node Characteristics

Provide the following information:

- Network ID

  This is the NETID of the network on which your workstation resides.

- Local node name

  This is the node name for your workstation. It can be any name you choose, provided that it is unique within your network. If you are not using Advanced Peer-to-Peer Networking (APPN), we suggest using your PU, but be aware that, if you use your PU, you should also define a local LU.

- Node type

  Select End node - no network node server.

- Local node ID

  This is the node ID for your workstation. The first box is the IDBLK, and the second is the IDNUM.

2.2.1.2 Defining a Local LU

Strictly speaking, you do not have to create a local LU to conduct CPIC conversations in a CM/2 environment. However, we recommend that you use local LUs (see also 2.2.1.7, “Specifying a Default Local LU” on page 22 for information on how to specify a default local LU).

Double-click on the Local LUs selection in the Features list box in the SNA Features List window (see Figure 10 on page 17).

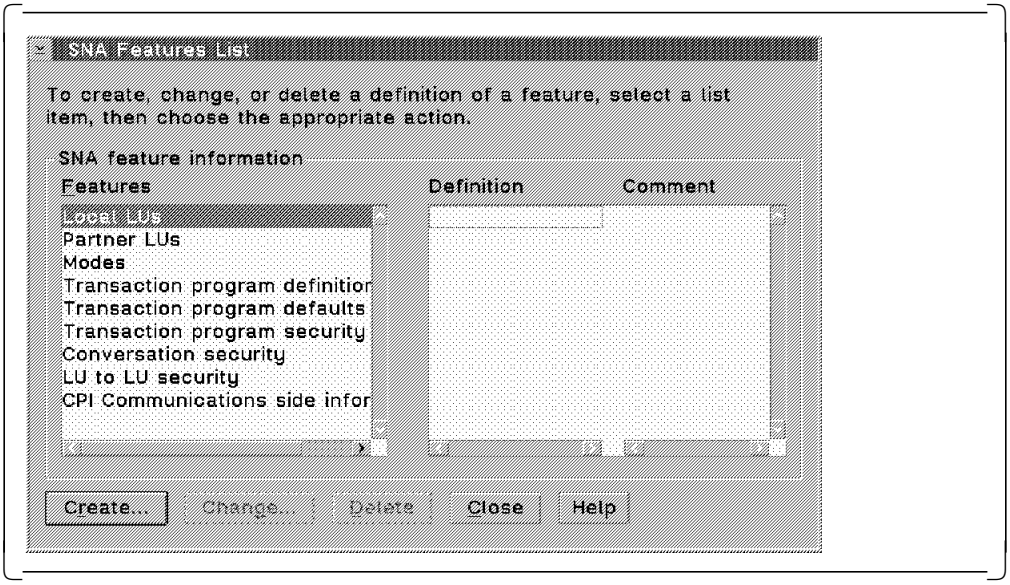


Figure 10. SNA Features List

The Local LU window is displayed (see Figure 11).

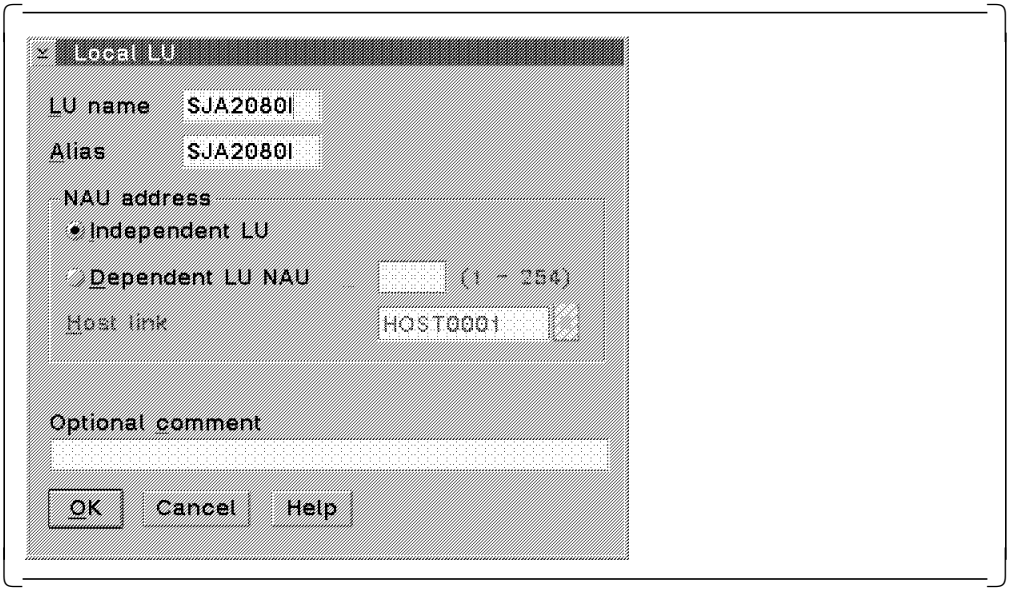


Figure 11. Local LU Definition

Provide the following information:

- LU name

  You need to know the name of an LU 6.2 enabled (independent) LU that is defined for your PU in VTAM. Your VTAM or network administrator should be able to tell you what your independent LUs are.

- Alias

  CM/2, DDCS for OS/2, and other applications that use APPC use this alias to refer to this particular local LU.

2.2.1.3 Defining a Connection to MVS

Once you have defined your local node, you must specify to CM/2 the destination address of your MVS system. Use the Connection to a Host window (see Figure 12) to define a connection to MVS.

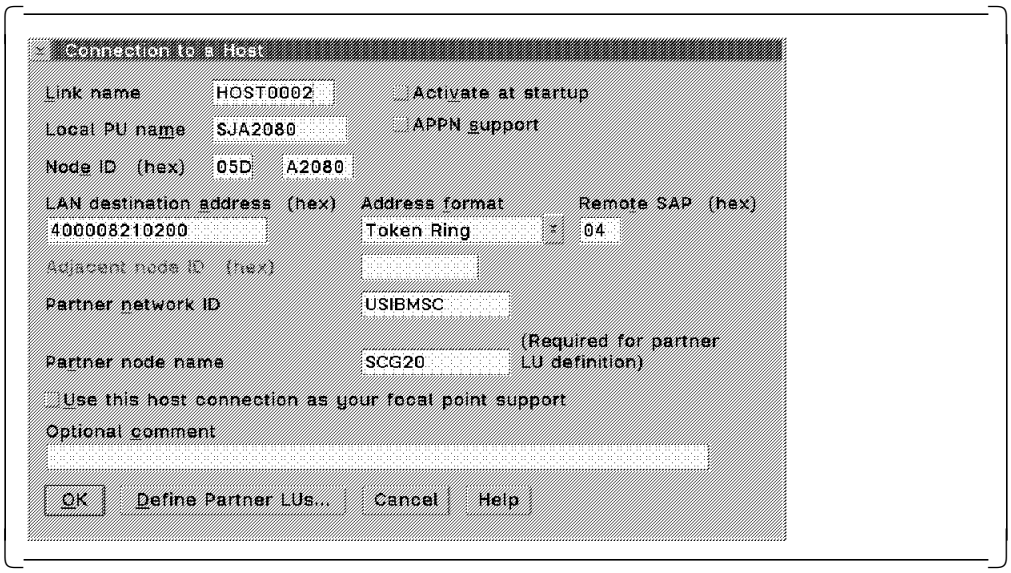


Figure 12. Connection to a Host

Provide the following information:

- Link name

  A unique name (within CM/2 on your workstation) for the connection between your workstation and MVS. CM/2 automatically generates the link name for you, but you can overtype it with a more meaningful link name.

- Activate at startup

  This toggle determines whether the connection between your workstation and the host is activated when CM/2 is started. We recommend that you not check this check box.

- APPN support

  Do not check this check box unless you are using APPN.

- Local PU name

  This is the PU name of your workstation.

- Node ID

  This is your node ID. If you have completed the Local Node Characteristics window (see Figure 9 on page 16), the values defined there are displayed here.

- LAN destination address

  This is the token-ring address of the 37XX controller.

- Partner network ID

  This is the NETID of the 37XX system. Often, but not always, it is the same NETID as the NETID you defined in the Local Node Characteristics window.

- Partner node name

  This is the address of your MVS system (or, more correctly, the address of the System Services Control Program (SSCP) for VTAM). Your MVS or network administrator should know what this address is.

2.2.1.4 Defining a Partner LU for MVS

Once you have defined the physical connection between your workstation and MVS, you must define at least one partner LU. Select the Define Partner LUs... push button in the Connection to a Host window to display the Partner LUs window (see Figure 13).

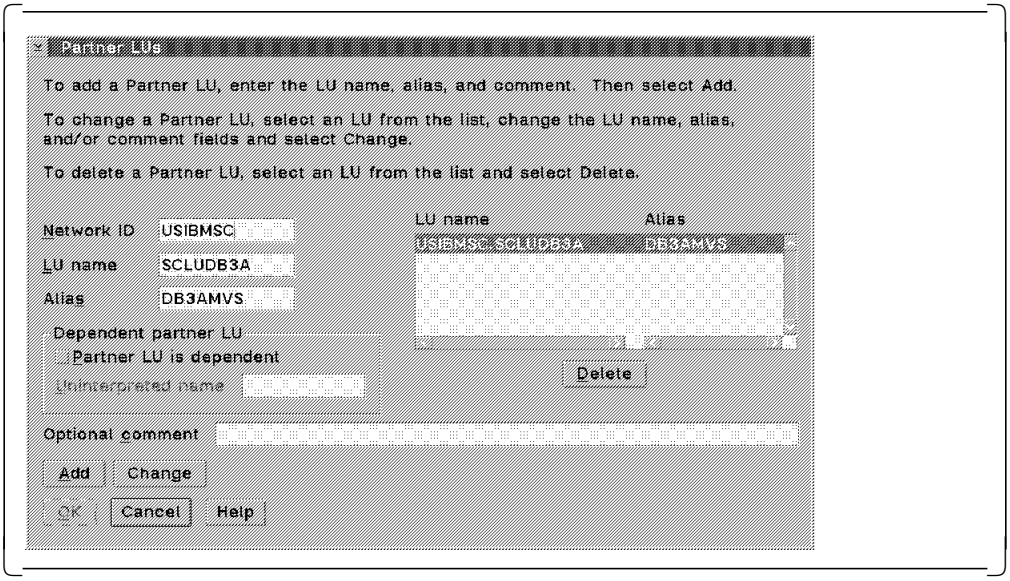


Figure 13. Partner LUs Definition

Provide the following information:

- Network ID

  This is the NETID where your MVS system resides.

- LU name

  This is the LU name at which the DB2 subsystem is defined in the APPL macro in VTAM.

- Alias

  This is the name by which this partner LU is referenced within CM/2 and other SNA applications. We recommend that you pick an alias that describes the partner LU, for example, DB3AMVS.

After filling in the above fields, select the Add push button to add an entry in the scrollable list box on the right-hand side of the window. Then select the OK push button.

2.2.1.5 Defining a Mode

A mode is used to define the conditions under which a transaction is run. IBMRDB is the mode name that is traditionally used for DRDA applications. However, this mode name is not defined by default to CM/2, so you must create a mode definition yourself.

In the SNA Features List (Figure 10 on page 17), double-click on the Modes selection in the Features list box to display the Mode Definition window (see Figure 14).

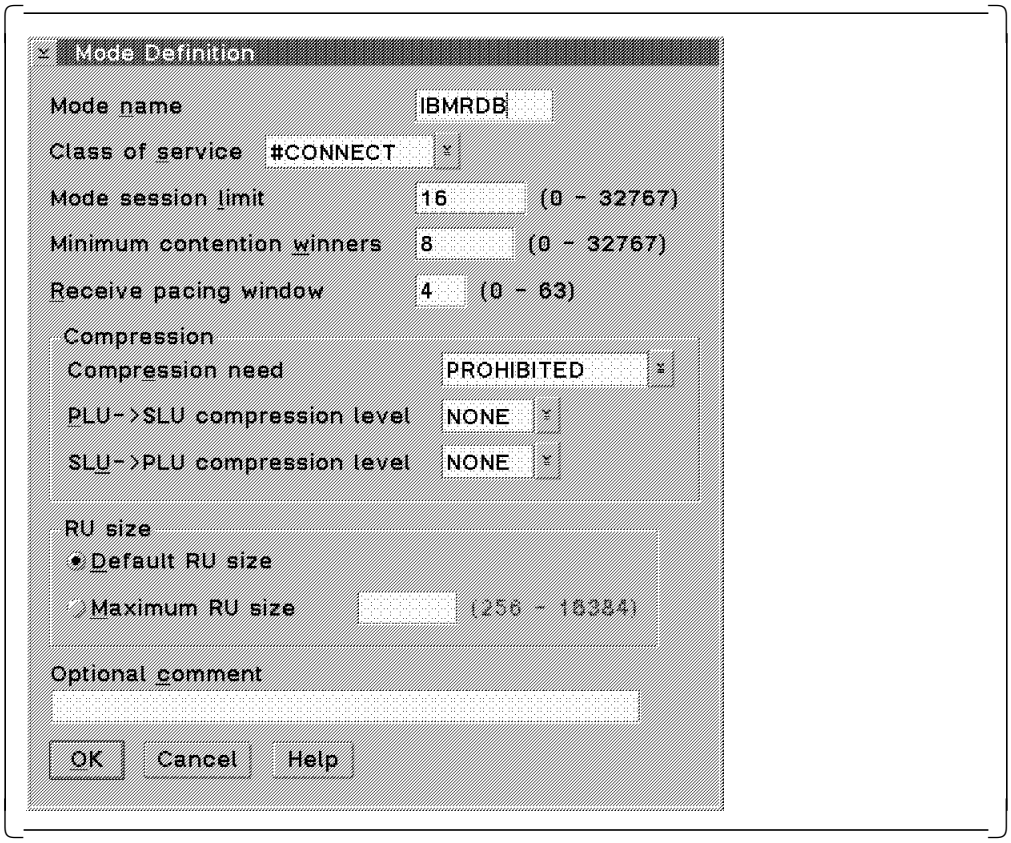


Figure 14. Mode Definition

Provide the following information:

- Mode name

  The mode name should be an available mode that is defined for DB2 in VTAM. Often, the mode name that is used is IBMRDB; however, it is not mandatory. Ask your VTAM administrator for a valid mode name.

- Class of service

  COS defines some data transmission characteristics, such as security, priority, and speed of the transmission. The COS that you select depends on the behavior that you want to implement. #CONNECT is a COS that is commonly used.

- Mode session limit

  The mode session limit controls the maximum number of *concurrent* sessions between two LUs at any time.

- Minimum contention winners

  In the Minimum contention winners field, you specify how many times your LU should win when there is contention with a partner LU. The value should be less than or equal to the mode session limit value (we recommend a value that is one-half the mode session limit value). If you use a value of zero, CM/2 will determine the value dynamically at session allocation time.

2.2.1.6 Defining Side Information

DDCS for OS/2 now uses CPIC side information to indicate the partner LU, TPN, and mode definitions for the remote database. Essentially, a CPIC side information defines the partner LU and the TPN that should be invoked at the partner location. In previous versions of DDCS for OS/2, this information was cataloged in the Database Connection Services (DCS) and node (workstation) directories.

From the SNA Features List (Figure 10 on page 17), double-click on CPI Communications side information to display the CPI Communications Side Information window (see Figure 15).

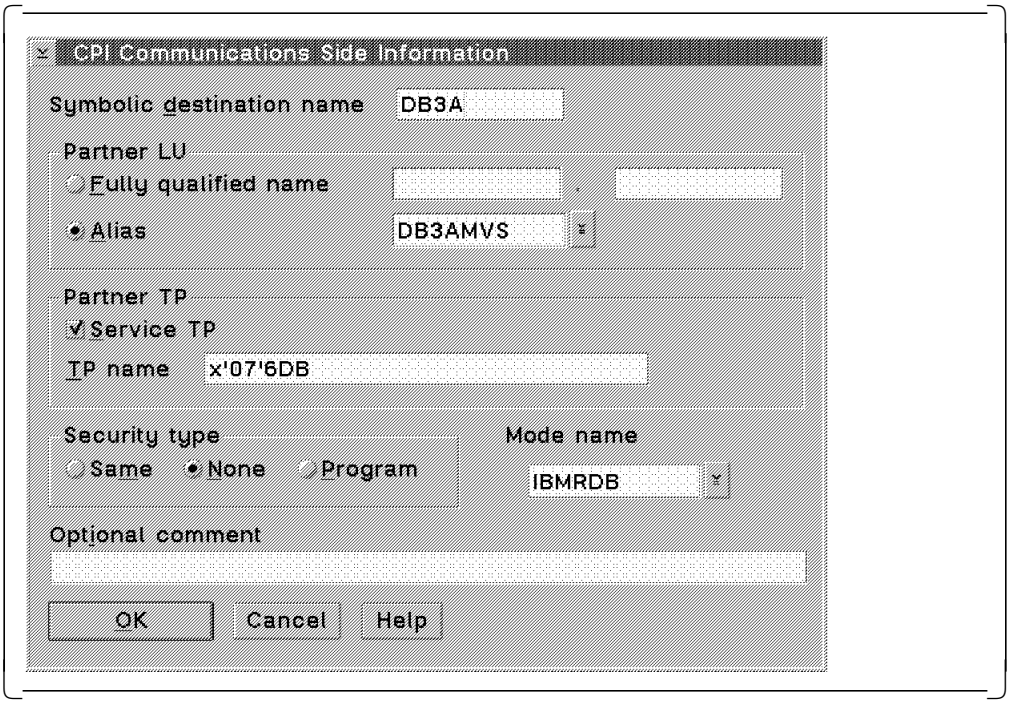


Figure 15. CPI Communications Side Information

Provide the following information:

- Symbolic destination name

  This is your name for the details that are stored in this particular CPIC side information. It can be any name of eight characters or less, and it is case sensitive. We recommend that you select a name that represents the target database name. The symbolic destination name is referenced when you catalog a node.

- Alias

  This is the partner LU alias that you entered when defining your partner LU. The Alias drop-down list box provides you with a list of the partner LU aliases that you have defined to CM/2.

- Service TP

  Check this check box. It specifies that the TPN that is run at the other end is a service TPN.

- TP name

  This is the name of the transaction program that runs when the session is established between your workstation and MVS. Enter x'07'6DB in this field.

- Security type

  This is the type of security that is used when CM/2 establishes the session. However, at run-time, DDCS for OS/2 will override whatever value you select here with the security specified in the node directory. You can select any of the three values, but we recommend that you select None.

- Mode name

  This is the mode that is used to determine the session characteristics. From the Mode name drop-down list box, select the mode that you cataloged in the Mode Definition window (Figure 14 on page 20).

2.2.1.7 Specifying a Default Local LU

You may have noticed that the CPI Communications Side Information window does not have a field that specifies which local LU is to be used in the connection. DDCS V2.3 uses CPIC side information combined with the default local LU, which you can define by using one of the following methods:

- APPCLLU OS/2 environment variable
- DEFAULT_LOCAL_LU_ALIAS(luname) parameter in CM/2
- Local node name.

CM/2 first checks for the APPCLLU OS/2 environment variable. If APPCLLU has not been set, it checks to determine whether the DEFAULT_LOCAL_LU_ALIAS parameter has been set in CM/2. If the parameter has not been set, CM/2 sets the default local LU to be the same as the local node name.

Note: If CM/2 uses the local node name as the default local LU, you must ensure that the local node name is *not* defined as a PU in the VTAM that is used when connecting to the host. If the local node name is defined as a PU, you will receive an APPC error. For this reason, we strongly recommend that you define a default local LU.

Let us assume that your CM/2 configuration is called 3270LAN, and that you want to set your default local LU to SJA2080I. To set the APPCLLU environment variable, you must add the following line to your CONFIG.SYS file:

```
SET APPCLLU=SJA2080I
```

You must reboot the machine for this change to become effective.

To set the DEFAULT_LOCAL_LU_ALIAS in CM/2, be sure that the 3270LAN.NDF file in the \CMLIB directory includes the line highlighted below:

```
DEFINE_DEFAULTS  IMPLICIT_INBOUND_PLU_SUPPORT(YES)
                 DEFAULT_MODE_NAME(BLANK)
                 DEFAULT_LOCAL_LU_ALIAS(SJA2080I)
                 MAX_MC_LL_SEND_SIZE(32767)
                 DIRECTORY_FOR_INBOUND_ATTACHES(*)
                 DEFAULT_TP_OPERATION(NONQUEUED_AM_STARTED)
                 DEFAULT_TP_PROGRAM_TYPE(BACKGROUND)
                 DEFAULT_TP_CONV_SECURITY_RQD(NO)
                 MAX_HELD_ALERTS(10);
```

You can place this line anywhere in the DEFINE_DEFAULTS section, provided that it is before the semicolon.

If you change the 3270LAN.NDF file, you must manually reverify by typing:

```
CMVERIFY 3270LAN /e
```

or by using the CMSETUP GUI.

Note: In CM/2 V1.11 you can specify which local LU is to be your default local LU by checking a check box in the Local LU window (see Figure 16).

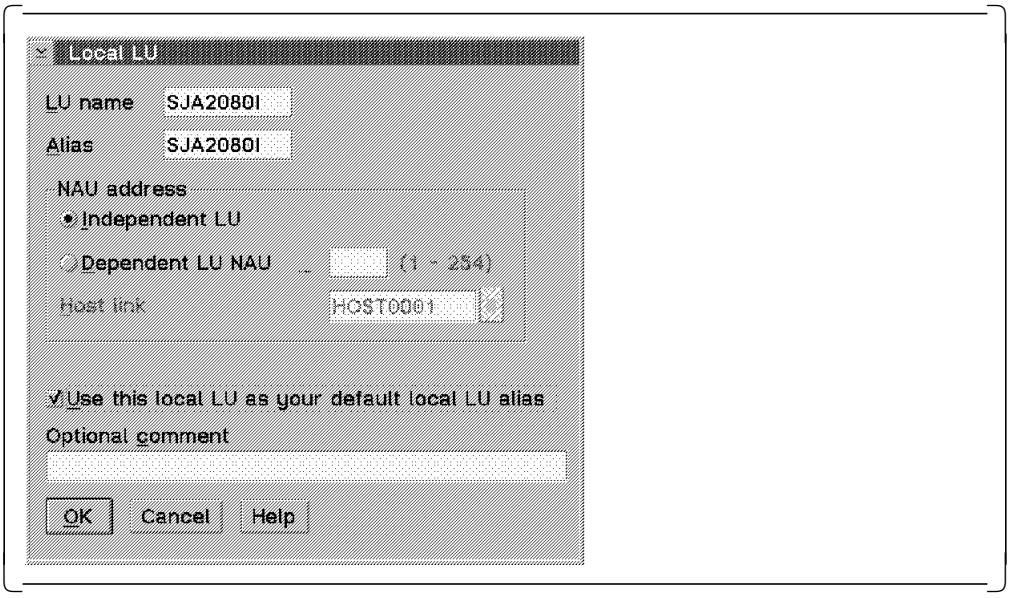


Figure 16. CM/2 V1.11 Local LU

2.2.1.8 Testing Your CM/2 Definitions

Once you have successfully saved and verified all of the CM/2 definitions, you should test that you can actually establish an APPC connection between the workstation and the host. The best way to test the connection is to use a program called Subsystem Management and perform the following steps:

1. Double-click on the Communications Manager/2 folder.
2. Double-click on the Subsystem Management icon.
3. Double-click on the SNA Subsystem item.
4. Double-click on the LU 6.2 sessions item.
5. Click on the Establish pull-down menu.
6. Click on the Establish... menu item.

Those actions display the Establish LU 6.2 session window (see Figure 17 on page 24).

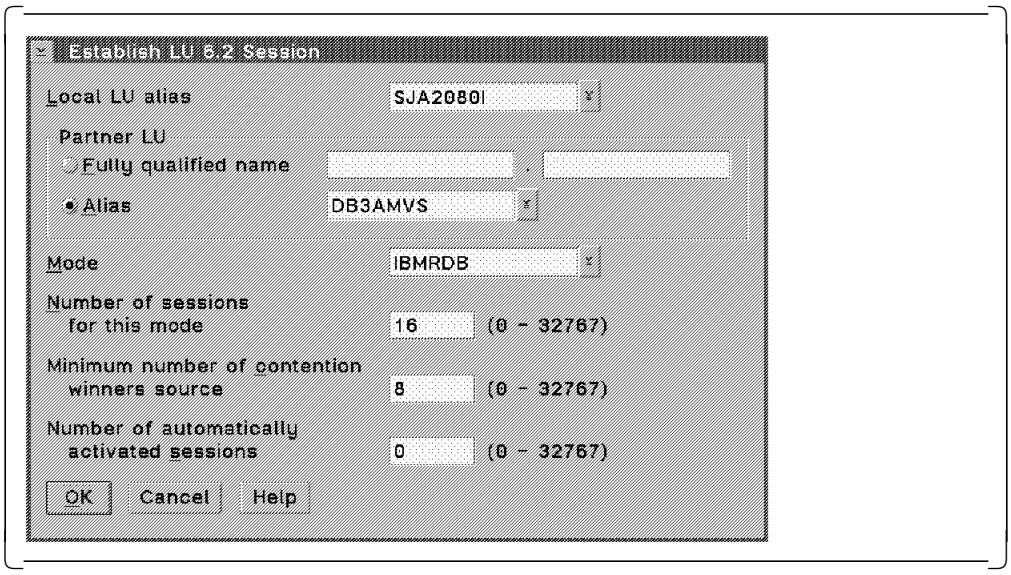


Figure 17. Establish LU 6.2 Session

Provide the following information:

- Local LU alias

  Use the drop-down list box to select the local LU alias that you specified as your default local LU.

- Alias

  Use the drop-down list box to select the partner LU alias that you cataloged when you were defining your partner LUs.

- Mode

  Use the drop-down list box to select the mode that you cataloged in CM/2.

Select the OK push button to establish the link. If all goes well, Subsystem Management displays the LU 6.2 Sessions window, which looks similar to Figure 18.

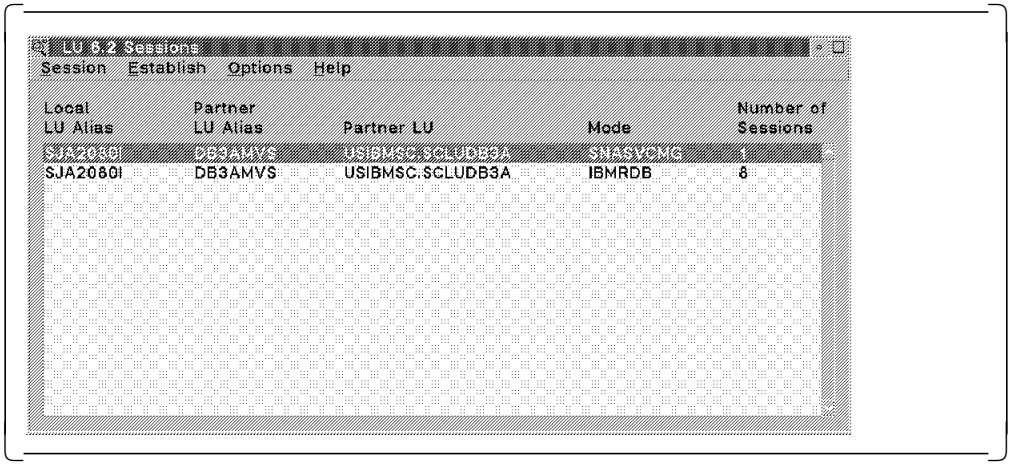


Figure 18. LU 6.2 Sessions

2.2.2 Configuring DDCS for OS/2

In order for DDCS for OS/2 to establish DRDA connections, you must complete several definitions by using database director or the command line. To start database director, double-click on the Database Director icon in the IBM DATABASE 2 folder. Refer to 2.6, "Updating Database Manager Configuration for DRDA AR" on page 44 for detailed information on how to catalog a DCS entry, a database, and a node.

2.2.3 Post-Setup Activities

To establish a DRDA connection, all you have to do is configure CM/2 and DDCS for OS/2 definitions on the workstation. Before you can begin using the DRDA connection in applications, however, you must perform several other activities.

2.2.3.1 Ensure That DB2 for MVS/ESA AS Is Configured

You must complete several tasks on the MVS host to enable applications to connect from the workstation to the host. Refer to 2.8, "DB2 for MVS/ESA in DRDA Environment" on page 51 for further information on these tasks.

2.2.3.2 Test DRDA Connection

Once you have configured both the workstation and the host, you can test the DRDA connection. The new form of the CONNECT statement used to connect to a host database is:

```
DB2 CONNECT TO database USER userid USING password
```

The database is the system database alias that you cataloged (see 2.6.2, "Catalog System Database" on page 46). The userid and password combination should be a valid MVS userid and password for the MVS system to which you are connecting.

2.2.3.3 Bind Packages to DB2 for MVS/ESA

After a successful connection, you must bind some packages into the DB2 for MVS/ESA database. A package is a database object that executes SQL statements against DB2 tables on behalf of a program.

Binding packages to MVS

```
db2 bind @ddcsmvs.lst blocking all grant public
```

The above command replaces the SQLJBIND program that was distributed with previous versions of DB2.

For a detailed description of the binding tasks and the authorities required, see Chapter 3, "Binding" on page 59.

2.3 Configuring DB2 for OS/2 As a DRDA AS

In this section, we outline the steps to define and establish a DB2 for OS/2 environment that supports incoming DRDA AR requests. The steps are:

1. Configure CM/2.
   a. Define a local node, connection to MVS, and partner LU.
   b. Define a TPN.

c. Start the attach manager.

d. Specify conversation security.

2. Configure DB2 for OS/2

   a. Update database director.

   b. Define APPC as an allowed protocol.

3. Carry out post-setup activities.

   a. Ensure that the DB2 for MVS/ESA DRDA AR is configured correctly.

   b. Test connection from DRDA AR.

   c. Bind packages to DB2 for OS/2.

2.3.1 Configuring CM/2

When a DRDA AR wants to connect to your DB2 for OS/2 workstation, it must do so through CM/2. Therefore, you must define certain features in CM/2 to allow inbound DRDA requests. In this section we cover those definitions.

2.3.1.1 Define a Local Node, Connection to MVS, and Partner LU

The procedure for defining these features are covered, respectively, in 2.2.1.1, “Defining a Local Node” on page 15; 2.2.1.3, “Defining a Connection to MVS” on page 18; and 2.2.1.4, “Defining a Partner LU for MVS” on page 19.

Note: You catalog a partner LU only if the workstation must to accept “already verified” connections. Refer to Chapter 4, “Security Considerations” on page 79 for further details on security.

2.3.1.2 Define a TPN

When CM/2 receives an incoming request, it extracts the TPN that is passed as part of the data and hands over control to the appropriate program (as specified in the CM/2 definition for that particular TPN). You define a TPN in the Transaction Program Definition window in CM/2 (see Figure 19). To get to the Transaction Program Definition window, double-click on Transaction program definition in the SNA Features List (see Figure 10 on page 17).

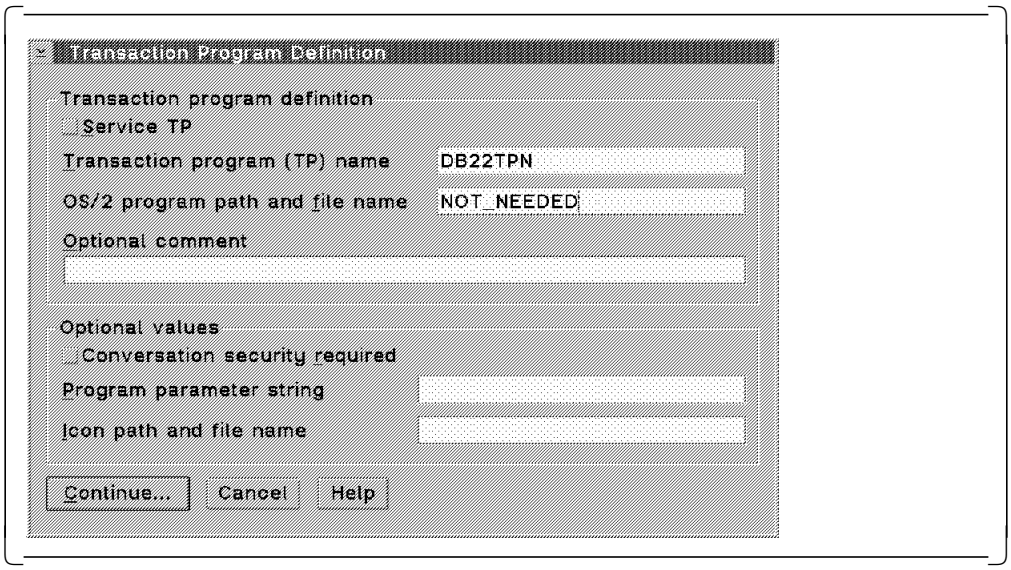


Figure 19. Transaction Program Definition

The TP name must match the name that is passed from the DRDA AR when requesting a DRDA connection (stored in the LINKATTR column of the SYSIBM.SYSLOCATIONS table in DB2 for MVS/ESA). Therefore we suggest that you choose a name that indicates the purpose of the TPN. The TP name entered here is also cataloged in the database manager configuration (see 2.7.1, “Registering TPN with DB2” on page 47).

The OS/2 program path and file name field contains the name of an executable OS/2 program that is executed when this TPN is invoked. The database manager automatically preloads the program that must be run when CM/2 accepts inbound DRDA requests, so CM/2 does not use this field. However, CM/2 does not let you continue if this field is blank, so you must enter some nonblank characters in this field to satisfy CM/2.

Click on the Continue... push button to complete the TPN definition (see Figure 20).

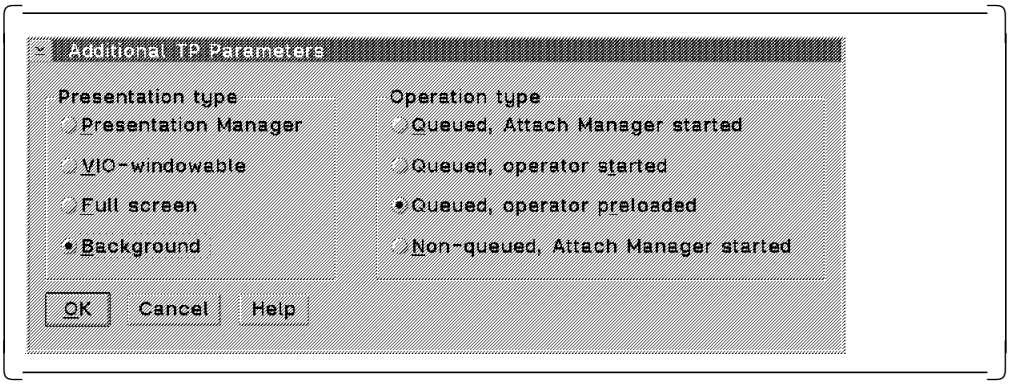


Figure 20. Additional TP Parameters

Select the program in the Background to indicate that it does not run under control of OS/2 Presentation Manager, and for the Operation type select Queued, operator preloaded to indicate that the program is started by DB2 for OS/2 when you issue the DB2START command.

Note: CM/2 must be started before DB2 for OS/2. If you stop CM/2, you must also stop and restart DB2 for OS/2 after you have restarted CM/2.

2.3.1.3 Start the Attach Manager

The CM/2 attach manager waits for incoming APPC connection requests and then passes the request to the appropriate TPN. If the attach manager is not started, all incoming APPC connection requests are refused.

To specify that the attach manager be started, use the Local Node Options window (see Figure 21 on page 28) and check the Activate Attach Manager at start up check box.

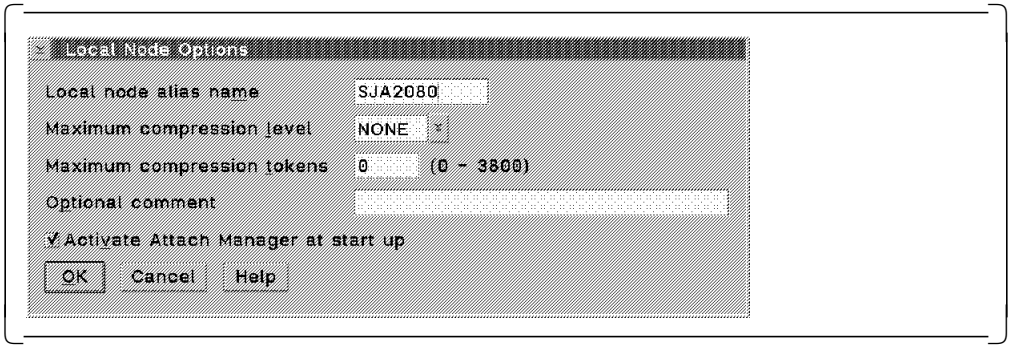


Figure 21. Starting Attach Manager

2.3.1.4 Specify Conversation Security

When DB2 for OS/2 receives an inbound DRDA request, it must ensure that the requester has the appropriate authorization. For authentication, you can either define userid and password combinations within CM/2 or pass the userid and password to User Profile Management (UPM) for authentication. We recommend that you use UPM for your security management.

To specify to CM/2 that you want to use UPM as your security manager, double-click on Conversation security in the SNA Features List (see Figure 10 on page 17). The Conversation Security window shown in Figure 22 is displayed.

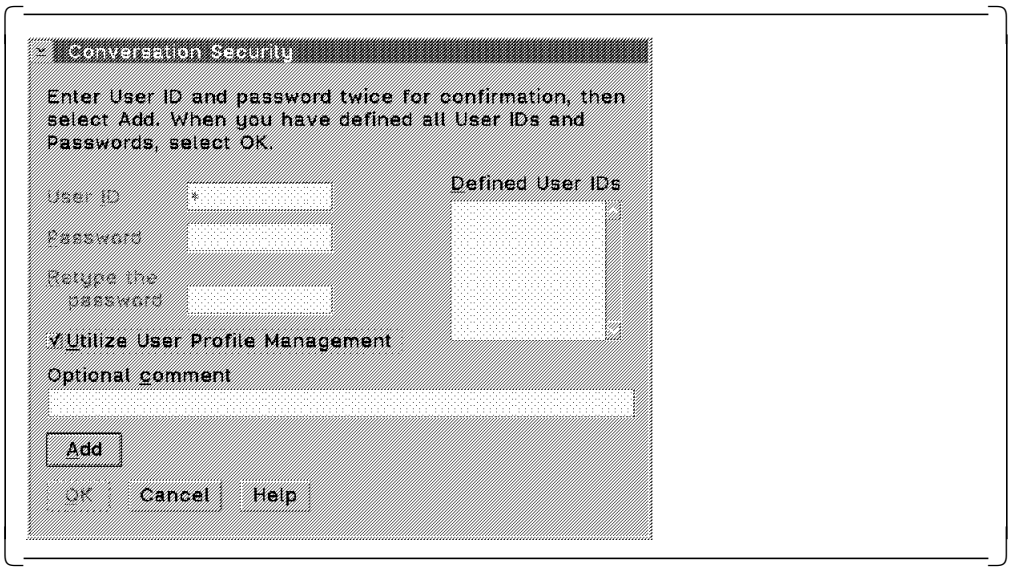


Figure 22. Using UPM as Security Manager

Check the Utilize User Profile Management check box. CM/2 disables all of the input entry fields, after it places an asterisk in the User ID field. You can also create profiles for specific userid and password combinations. CM/2 first attempts to find a matching profile for the userid received in the message. If it finds a matching profile, authentication is performed within CM/2. If it does not find a matching profile, it checks for the existence of a profile configured with an asterisk to invoke UPM for authentication.

2.3.2 Configuring DB2 for OS/2

On receiving an incoming DRDA connection request, CM/2 determines which TPN should be invoked, validates the incoming userid and password, and then passes control to DB2 for OS/2. You must configure some parameters in DB2 for OS/2 so that CM/2 knows how to pass on the incoming request. Also, you must ensure that the details for the target database are cataloged.

You must also update the CONFIG.SYS file to specify that APPC is an allowed protocol.

2.3.2.1 Update Database Director

To start database director, double-click on the Database Director icon in the IBM DATABASE 2 folder. Refer to 2.7, "Updating Database Manager Configuration for DRDA AS" on page 47 for detailed information on how to update the necessary fields in database director.

2.3.2.2 Define APPC As an Allowed Protocol

DB2 for OS/2 can use several communications protocols when accepting clients. To accept a DRDA client, DB2 for OS/2 must use the APPC protocol. To specify that APPC is an allowed protocol, the DB2COMM OS/2 environment variable must be set. Add the following line to the CONFIG.SYS file of the workstation that is acting as the DRDA AS:

```
SET DB2COMM=APPC
```

Note: If more than one protocol is to be supported, the protocols must be comma-delimited; for example:

```
SET DB2COMM=APPC,NETBIOS,TCPIP
```

2.3.3 Post-Setup Activities

The CM/2 and DB2 for OS/2 definitions are the only definitions that you must configure on the workstation to receive an incoming DRDA connection. However, before you can begin using the DRDA connection in applications, you must perform several other activities.

2.3.3.1 Ensure That DB2 for MVS/ESA AR Is Configured

You must complete several tasks on the MVS host to enable applications to connect to the workstation from the host. Refer to 2.8, "DB2 for MVS/ESA in DRDA Environment" on page 51 for more information on these tasks.

2.3.3.2 Test DRDA Connection

Once you have configured both CM/2 and DB2 for OS/2, you can test the connection. The steps to test this connection are covered in 2.8, "DB2 for MVS/ESA in DRDA Environment" on page 51.

Note: Depending on how your VTAM definitions have been set up, you may have to activate the link from the workstation to the host and establish an LU 6.2 session over the link. Both of these tasks are accomplished by using Subsystem Management. By manually activating the link, and establishing a session, you create an entry in a VTAM table called ISTPDILU, which VTAM can then reference for the connection from the host to the workstation.

2.3.3.3 Bind Packages to DB2 for OS/2

After a successful connection, you must bind some packages into the DB2 for OS/2 database. A package is a database object that executes SQL statements against DB2 tables on behalf of a program. Refer to Chapter 3, “Binding” on page 59 for a detailed description of the tasks and authorities involved in binding packages.

2.4 Configuring DDCS for AIX As a DRDA AR

In this section we outline the steps to define and establish a DRDA connection using DDCS for AIX from an AIX workstation to an MVS host. The steps are:

1. Configure SNA Server/6000

   DDCS for AIX has not changed in terms of how it uses the SNA subsystem. In other words, it still uses LU 6.2 side information to specify the remote target.

2. Configure DDCS for AIX

   The new release of DDCS for AIX has a completely new GUI for the directories. The new GUI, called database director, provides the same function as the previous db2adm program but has a different presentation interface.

3. Carry out post-setup activities

   Other tasks you must perform after you complete the SNA Server/6000 and DDCS for AIX configurations include ensuring that the DB2 for MVS/ESA system has been configured correctly and binding packages to the remote database.

2.4.1 Configuring SNA Server/6000

The DRDA protocol uses APPC to establish connections to remote DB2 servers. SNA Server/6000 provides the interface to APPC in the AIX environment. In order for SNA Server/6000 to manage your APPC environment, you must define several profiles. To update these profiles, use the System Management Interface Tool (SMIT). To display the initial SNA Server/6000 panel, type `smit sna` on the AIX command line.

Note: You must be logged in as root to update SNA definitions.

2.4.1.1 Define a Local Node

First you must define the attributes of your workstation. These attributes may already have been defined when SNA Server/6000 was installed. If not, use the Initial Node Setup panel in SMIT (see Figure 23 on page 31).

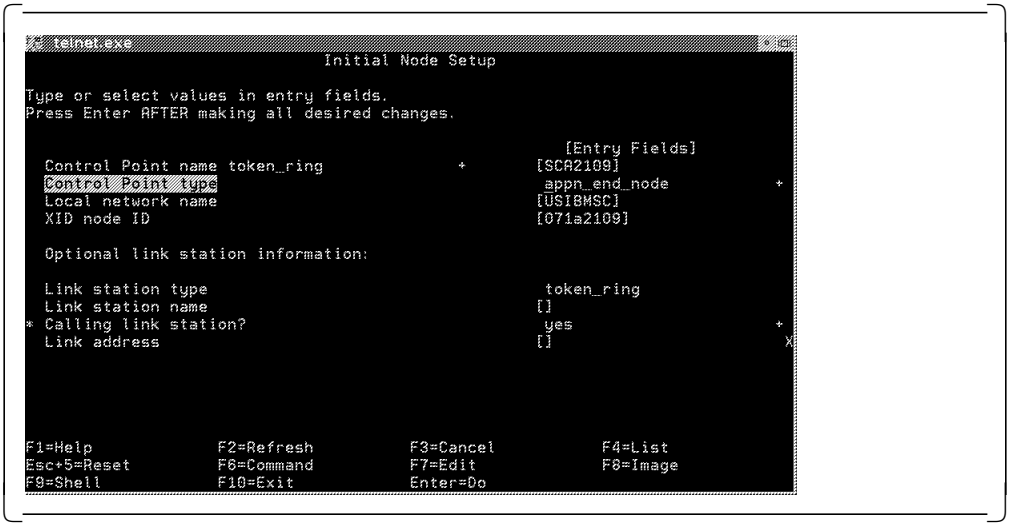


Figure 23. Initial Node Setup

Provide the following information:

- Control Point name

  The control point name can be any value you choose, provided that it is unique within your network. We suggest using your PU name.

- Control Point type

  The control point type defines your workstation node. Typically it is an appn_end_node.

- Local network name

  The local network name is the NETID of the network where your workstation resides.

- XID node ID

  XID node ID is the node ID for your workstation. The first three characters are the IDBLK, and the remaining characters are the IDNUM. This value is used in the XID exchange with your partner LUs.

- Link station type

  Link station type defines the physical network used to communicate with your partner LUs.

2.4.1.2 Define a Local LU

A local LU is used in the LU 6.2 side information to specify which LU in your workstation is used in the connection to the host.

Select the LU 6.2 Local LU option from the LU 6.2 panel and add a profile to display the Add LU 6.2 Local LU Profile (see Figure 24 on page 32).

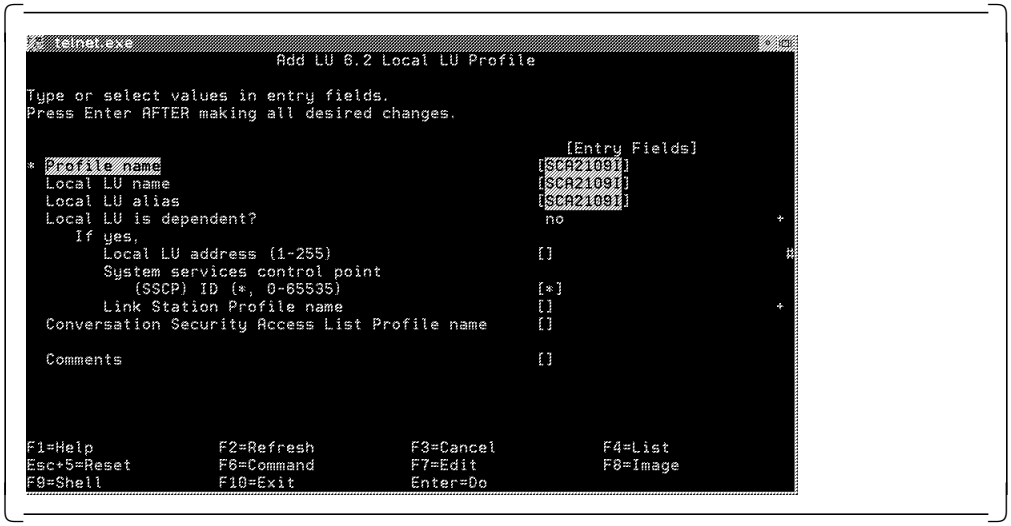


Figure 24. Add LU 6.2 Local LU Profile

Provide the following information:

- Profile name

  Profile name is the name of the SNA Server/6000 profile that contains your local LU 6.2 details.

- Local LU name

  Local LU name is the LU that is defined in VTAM as being an independent LU belonging to your PU.

- Local LU alias

  Local LU alias is the alias by which this local LU is known within SNA Server/6000.

2.4.1.3 Defining a Connection to MVS

Once you have defined a local node, you must specify to SNA Server/6000 the destination address of your MVS system. Specifying the destination address is achieved by using two seperate panels. On the Token Ring Link Station panel, you define the physical characteristics of the connection (see Figure 25 on page 33 and Figure 26 on page 33), and on the Partner LU 6.2 Location panel, you define the logical characteristics of the connection (see Figure 27 on page 34).

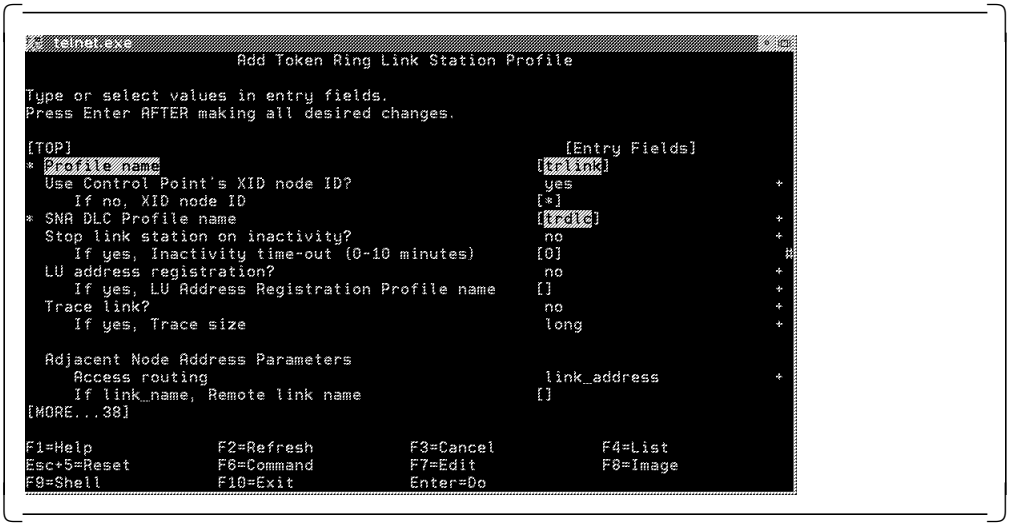


Figure 25. Add Token Ring Link Station Profile: Page 1

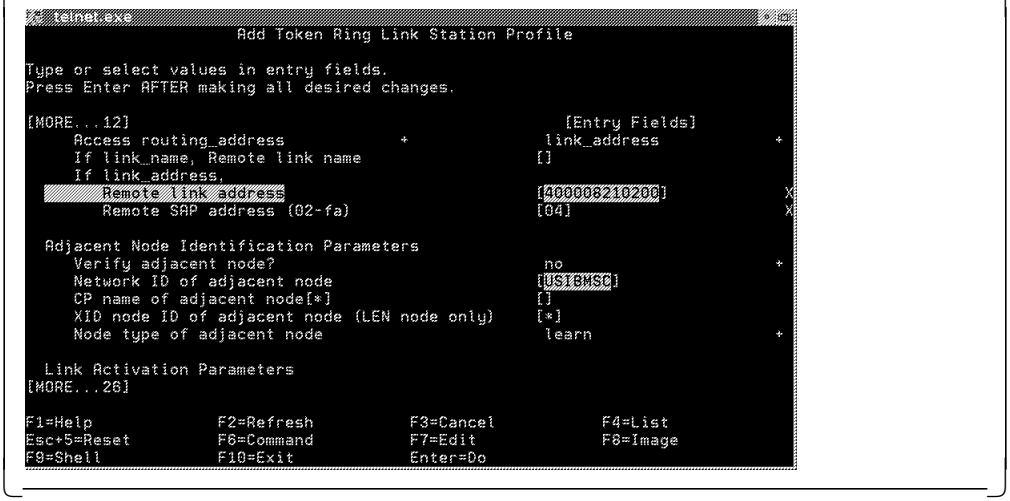


Figure 26. Add Token Ring Link Station Profile: Page 2

Provide the following information:

- Profile name

  Profile name is the name of the SNA Server/6000 profile that contains your Token Ring Link Station details.

- SNA DLC Profile name

  This is the profile name of the DLC that you use when establishing a connection.

- Access routing

  Access routing should be link_address.

- Remote Link address

  Remote link address should be the TIC (token ring address) of the 37XX controller that is used to connect to the host.

- Remote SAP address

  Remote SAP address should be 04.

- Network ID of adjacent node

  The network ID of the adjacent node is the NETID of the 37XX controller that is used to connect to the host.

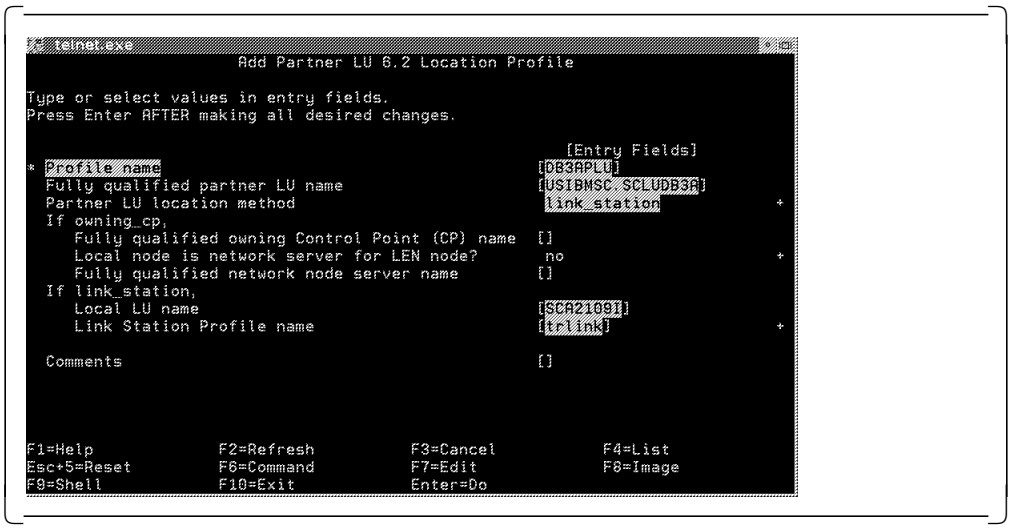


Figure 27. Add Partner LU 6.2 Location Profile

Provide the following information:

- Profile name

  Profile name is the name of the SNA Server/6000 profile that contains your partner LU 6.2 location details.

- Fully qualified partner LU name

  A fully qualified partner LU name consists of your host's NETID and LU name concatenated with a period between them.

- Partner LU location method

  Partner LU location method specifies the route that SNA Server/6000 will take when establishing a session with the partner LU. Specify link_station, unless you are using APPN.

- Local LU name

  The local LU name is name of your local LU.

- Link Station Profile name

  The link station profile name is the profile name of the link station that is used when connecting to the host (see Figure 25 on page 33).

2.4.1.4 Define a Partner LU for MVS

Once you have defined the physical connection between your workstation and MVS, you must define at least one partner LU.

Select the LU 6.2 Partner LU option from the LU 6.2 panel and add a profile to get to the Add LU 6.2 Partner LU Profile (see Figure 28 on page 35).

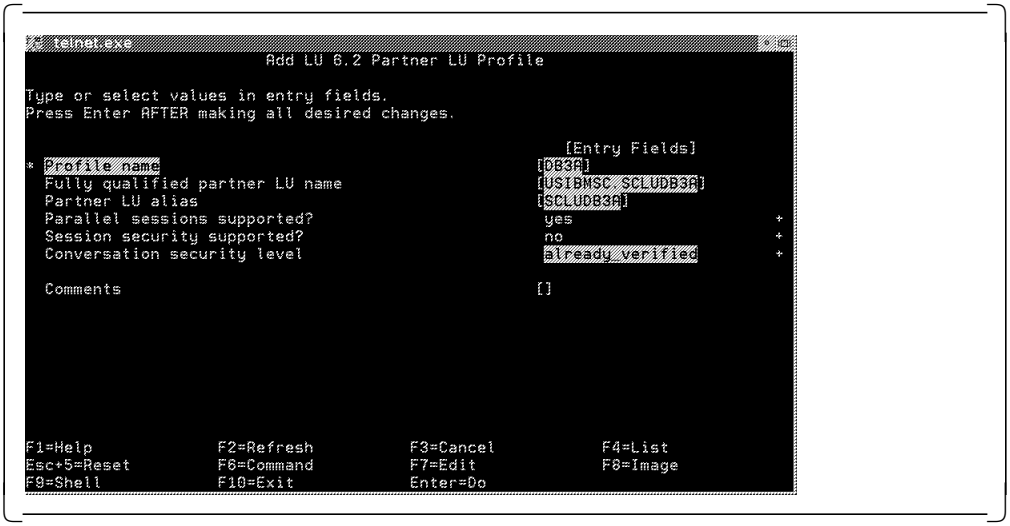


Figure 28. Add LU 6.2 Partner LU Profile

Provide the following information:

- Profile name

  Profile name is the name of the SNA Server/6000 profile that contains your partner LU details.

- Fully qualified partner LU name

  A fully qualified partner LU name consists of your host's NETID and LU name concatenated with a period between them. The LU name is the LU name that the DB2 subsystem defined to VTAM in the APPL macro.

- Partner LU alias

  The partner LU alias is the name by which this partner LU is referenced within SNA Server/6000. We recommend that you pick an alias that describes the partner LU.

- Conversation security level

  Conversation security level normally specifies the level of security used when establishing a conversation to this partner LU. DDCS for AIX does not use this information when the workstation is acting as a DRDA AR.

2.4.1.5 Define a Mode

A mode is used to define the conditions under which a transaction is run. IBMRDB is the mode that is traditionally used for DRDA applications. However, this mode name is not defined by default to SNA Server/6000, so you must create a mode entry yourself.

Select the LU 6.2 Mode option from the LU 6.2 panel and add a profile to get to the Add LU 6.2 Mode Profile (see Figure 29 on page 36).

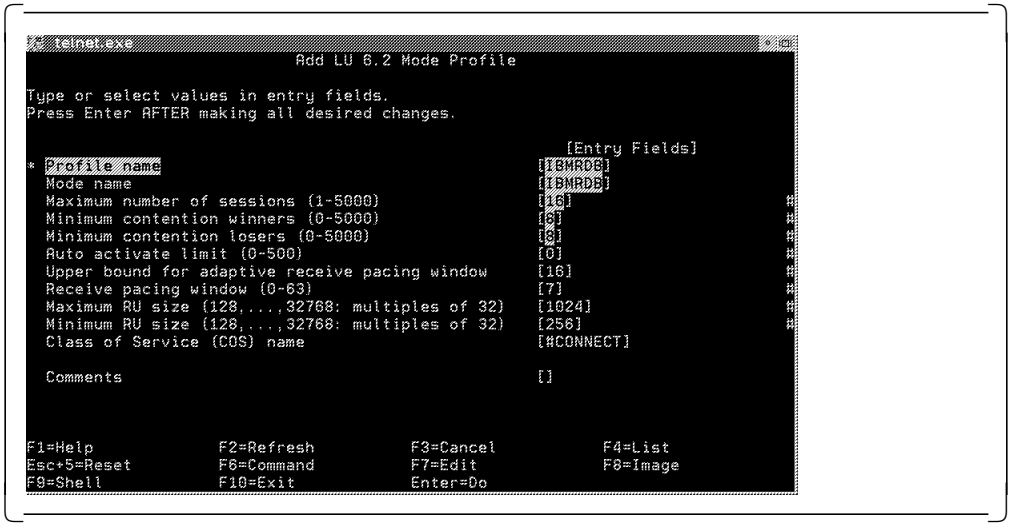


Figure 29. Add LU 6.2 Mode Profile

Provide the following information:

- Profile name

  The profile name is the name of the SNA Server/6000 profile that contains your mode details.

- Mode name

  The mode name should be an available mode that is defined for DB2 in VTAM. Often, the mode name that is used is IBMRDB; however it is not mandatory. Talk to your VTAM administrators to find out a valid mode name.

- Maximum number of sessions

  The maximum number of sessions parameter controls the maximum number of *concurrent* sessions between two LUs at any time.

- Minimum contention winners

  Minimum contention winners specifies how many times your LU should win when there is contention with a partner LU. The value should be less than or equal to the maximum number of sessions (we recommend a value that is one-half the maximum number of sessions).

- Class of service (COS) name

  COS defines some data transmission characteristics, such as security, priority, and speed of the transmission. The COS that you select is dependent on the behavior that you want to implement. #CONNECT is a COS that is commonly used.

2.4.1.6 Define Side Information

DDCS for AIX uses side information to indicate the partner LU, TPN, and mode definitions. Essentially, a side information profile defines the partner LU and the TPN that should be invoked at the partner location.

Select the LU 6.2 Side Information option from the LU 6.2 panel and add a profile to get to the Add LU 6.2 Side Information Profile (see Figure 30 on page 37).

Figure 30. Add LU 6.2 Side Information Profile

Provide the following information:

- Profile name

  Profile name is the name of the SNA Server/6000 profile that contains your side information details. The side information profile name is referenced when you catalog a node.

- Local LU or Control Point alias

  Local LU alias is the alias that you defined in Figure 24 on page 32.

- Partner LU alias

  Partner LU alias is the alias for your partner LU. You defined your partner LU alias in Figure 28 on page 35.

- Mode name

  Mode name is the mode that is used to determine the session characteristics. Choose the mode name that you defined in Figure 29 on page 36.

- Remote transaction program name (RTPN)

  RTPN is the name of the TP that runs when the session is established between your workstation and MVS. Enter 07f6c4c2 in this field (case is unimportant).

- RTPN in hexadecimal?

  RTPN in hexadecimal? specifies whether the RTPN is entered in hexadecimal. Enter yes.

2.4.1.7 Verify your SNA Server/6000 Definitions

Once you have updated all of the necessary SNA Server/6000 profiles, you must verify them to check for errors. To verify your profiles, select Verify Configuration Profiles from the Advanced Configuration panel in SMIT (see Figure 31 on page 38). When prompted for Update Action if verification successful, select dynamic_update and press Enter.

Figure 31. Verify Configuration Profiles

Note: You may need to stop and restart SNA Server/6000 in order for some changes to become effective.

2.4.1.8 Test Your SNA Server/6000 Definitions

Once you have successfully saved and verified all of the SNA Server/6000 profiles, you should test that you can actually establish an APPC connection between the workstation and the host.

To test the connection, perform the following steps from `smit sna` using the root (or equivalent) username:

1. Select Manage SNA Resources.
2. Select Start SNA Resources.
3. Select Start SNA.

These actions start the SNA subsystem (if it has not already been started). Once SNA has been started, perform the following steps:

1. Select Start an SNA Link Station.
2. Enter the token ring link station profile name you defined in Figure 25 on page 33.
3. Press Enter.

These actions activate the link between your workstation and the host. To establish an SNA session, select Start an SNA Session. This action displays the Start an SNA Session panel (see Figure 32 on page 39).

Figure 32. Start an SNA Session

Provide the following information:

- Local LU alias

  Select the local LU alias that you defined in the Add LU 6.2 Local LU Profile (see Figure 24 on page 32).

- Partner LU alias

  Select the partner LU alias that you defined in the Add LU 6.2 Partner LU Profile (see Figure 28 on page 35).

- Mode name

  Select the mode name that you defined in the Add LU 6.2 Mode Profile (see Figure 29 on page 36).

Press Enter to establish the link. If all goes well, SNA Server/6000 displays the COMMAND STATUS panel with the result (see Figure 33).

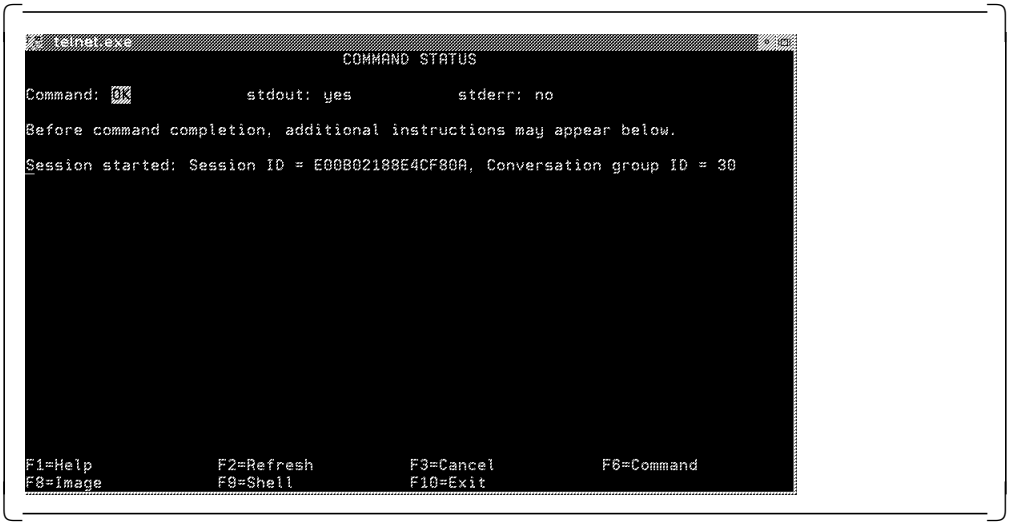


Figure 33. COMMAND STATUS

2.4.2 Configuring DDCS for AIX

In order for DDCS for AIX to establish DRDA connections, you must complete several definitions by using database director or the command line. To use database director, you must be in an X-Windows environment. Type db2dd at a command line prompt to start the database director. Refer to 2.6, "Updating Database Manager Configuration for DRDA AR" on page 44 for detailed information on how to catalog a DCS entry, a database, and a node.

2.4.3 Post-Setup Activities

To establish a DRDA connection, all you have to do is configure SNA/Server 6000 and the DDCS for AIX definitions. However, before you can begin using the DRDA connection in applications, you must perform several other activities.

2.4.3.1 Ensure That DB2 for MVS/ESA AS Is Configured

You must complete several tasks on the MVS host to allow applications to connect from the workstation to the host. Refer to 2.8, "DB2 for MVS/ESA in DRDA Environment" on page 51 for further information on these tasks.

2.4.3.2 Test DRDA Connection

Once you have configured both the workstation and the host, you can test the DRDA connection. The command to connect to a host database is:

```
db2 connect to database user userid using password
```

The database is the system database alias that you cataloged (see 2.6.2, "Catalog System Database" on page 46). The userid and password combination should be a valid MVS userid and password for the MVS system to which you are connecting.

2.4.3.3 Bind Packages to DB2 for MVS/ESA

After a successful connection, you must bind some packages into the DB2 for MVS/ESA database. A package is a database object that executes SQL statements against DB2 tables on behalf of a program.

Binding packages to MVS

```
db2 bind @ddcsmvs.lst blocking all grant public
```

The above command replaces the sqljbind program that was distributed with previous versions of DB2.

For a detailed description of the binding tasks and the authorities required, see Chapter 3, "Binding" on page 59.

2.5 Configuring DB2 for AIX As a DRDA AS

In this section we outline the steps to define and establish a DB2 for AIX environment that supports incoming DRDA AR requests. The steps are:

1. Configure SNA Server/6000.
   a. Define a local node, connection to MVS, and partner LU.
   b. Define a TPN.
   c. Specify conversation security.

2. Configure DB2 for AIX.
   a. Update database director.
   b. Define APPC as an allowed protocol.
3. Carry out post-setup activities.
   a. Ensure that the DB2 for MVS/ESA DRDA AR is configured correctly.
   b. Test connection from DRDA AR.
   c. Bind packages to DB2 for AIX.

2.5.1 Configuring SNA Server/6000

When a DRDA AR wants to connect to your DB2 for AIX workstation, it must do so through SNA Server/6000. Therefore, SNA Server/6000 must have certain profiles configured to allow inbound DRDA requests to be accepted. In this section, we cover the definitions that must be specified in SNA Server/6000.

2.5.1.1 Define a Local Node, Connection to MVS, and Partner LU

The procedure for defining these profiles are covered, respectively, in 2.4.1.1, “Define a Local Node” on page 30; 2.4.1.3, “Defining a Connection to MVS” on page 32; and 2.4.1.4, “Define a Partner LU for MVS” on page 34.

2.5.1.2 Define a TPN

When SNA Server/6000 receives an incoming request, it extracts the TPN that is passed as part of the data and hands control to the appropriate program (as specified in the SNA Server/6000 definition for that particular TPN).

Select the LU 6.2 Transaction Program Name (TPN) option from the LU 6.2 panel and add a profile to get to the Add LU 6.2 TPN Profile panel (see Figure 34).

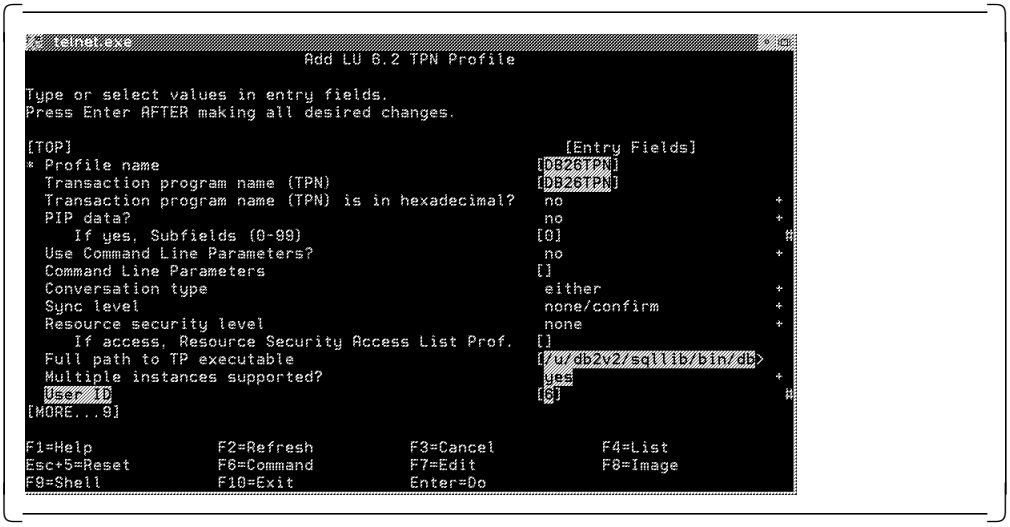


Figure 34. Add LU 6.2 TPN Profile

Provide the following information:

- Profile name

  Profile name is the name of the SNA Server/6000 profile that contains your LU 6.2 TPN details. The profile name entered here is cataloged in the database manager configuration (see 2.7.1, “Registering TPN with DB2” on page 47).

- Transaction Program Name (TPN)

  The TPN must match the name that is passed from the DRDA AR when requesting a DRDA connection (stored in the LINKATTR column of the SYSIBM.SYSLOCATIONS table in DB2 for MVS/ESA), so we suggest that you choose a name that indicates the purpose of the TPN.

- Transaction Program Name (TPN) is in hexadecimal?

  Enter no.

- Full path to TP executable

  The full path the TP executable is the fully qualified AIX path for the program that is to be executed when this TPN is invoked. The program that should be executed is db2acntp, which is normally located in the sqllib/bin directory of the DB2 instance owner (for example, `/u/db2v2/sqllib/bin/db2acntp`).

- Multiple instances supported?

  Multiple instances supported specifies whether multiple copies of the TPN can be executed concurrently.

- User ID

  The user ID is the unique numeric value for the DB2 instance owner. To find this value type `id` at the command line when logged in as the database instance owner. The user ID is the numeric value assigned to uid.

2.5.1.3 Specify Conversation Security

In an OS/2 environment, you must specify in CM/2 that UPM will be used for conversation security validation. However, in AIX, because the security mechanism is built into the operating system, there is no need to specify the security mechanism to SNA Server/6000.

2.5.1.4 Update SNA System Defaults

To allow incoming DRDA requests, you must add the primary group name of every DB2 instance owner to the trusted group names for SNA. To find the group name for a user, type `id` on a command line. The value in parentheses assigned to the gid field is the group name for that user.

In the Change/Show SNA Node Profile panel, add the group name to the trusted group names field (see Figure 35 on page 43).

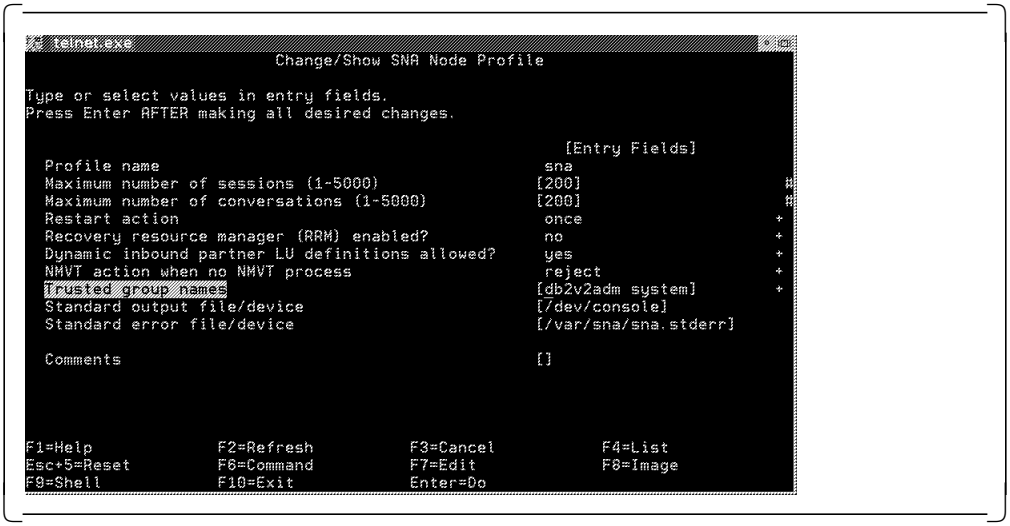


Figure 35. Updating SNA System Defaults

2.5.2 Configuring DB2 for AIX

On receiving an incoming DRDA connection request, SNA Server/6000 determines which TPN should be invoked, validates the incoming userid and password, and then passes control to DB2 for AIX. You have to configure some parameters in DB2 for AIX so that SNA Server/6000 knows how to pass on the incoming request. Also, you must ensure that the details for the target database are cataloged.

2.5.2.1 Update Database Director

To use database director, you must be in an X-Windows environment. Type db2dd at a command line prompt to start the database director. Refer to 2.7, “Updating Database Manager Configuration for DRDA AS” on page 47 for detailed information on how to update the necessary fields in database director.

2.5.2.2 Define APPC As an Allowed Protocol

DB2 for AIX can use several communications protocols when accepting clients. To accept a DRDA client, DB2 for AIX must use the APPC protocol. To allow this, you must set the AIX DB2COMM environment variable. The db2profile file that resides in the sqllib directory contains the environment variables that must be set. Using your favorite editor, append the db2profile to the .profile file of the user that will start the DB2 subsystem, or type the following command:

```
cat $HOME/sqllib/db2profile >> $HOME/.profile
```

2.5.3 Post-Setup Activities

To receive an incoming DRDA connection, all you have to do is configure SNA Server/6000 and the DB2 for AIX definitions. However, before you can begin using the DRDA connection in applications, you must perform several other activites.

2.5.3.1 Ensure That DB2 for MVS/ESA AR Is Configured

You must complete several tasks on the MVS host to allow applications to connect to the workstation from the host. Refer to 2.8, “DB2 for MVS/ESA in DRDA Environment” on page 51 for further information on these tasks.

2.5.3.2 Test DRDA Connection

Once you have configured both SNA Server/6000 and DB2 for AIX, you can test the connection. The steps involved in testing this connection are covered in 2.8, “DB2 for MVS/ESA in DRDA Environment” on page 51.

Note: Depending on how your VTAM definitions have been set up, you may have to activate the link from AIX to the host and establish an LU 6.2 session over the link by using SMIT sna. By doing so, you create an entry in a VTAM table called ISTPDILU, which VTAM can then reference for the connection from host to workstation.

2.5.3.3 Bind Packages to DB2 for AIX

After a successful connection, you must bind some packages into the DB2 for AIX database. A package is a database object that executes SQL statements against DB2 tables on behalf of a program. For a detailed description of the binding tasks and authorities required, see Chapter 3, “Binding” on page 59.

2.6 Updating Database Manager Configuration for DRDA AR

In this section, we cover the steps required to catalog a DCS entry, a system database entry, and a node entry. Because the AIX and OS/2 database director implementations are so similar, we cover only the OS/2 version.

You must stop and restart DB2 common server after making these changes.

2.6.1 Catalog Node

DDCS must know the destination of the DRDA connection. The information about the destination is stored in a side information, and the node directory specifies which side information will be used.

From within the database director perform the following steps:

1. Click on the sign next to the Database Managers icon.
2. Click on the sign next to the DB2 icon.
3. Click on the sign next to the Directories icon.
4. Click on the Node Directory icon with the right mouse button.
5. Select the Open as details menu item.
6. Select the Directory entry menu.
7. Select the Catalog... menu item.

These actions open the Node Directory Entry - Catalog notebook (see Figure 36 on page 45 and Figure 37 on page 45).

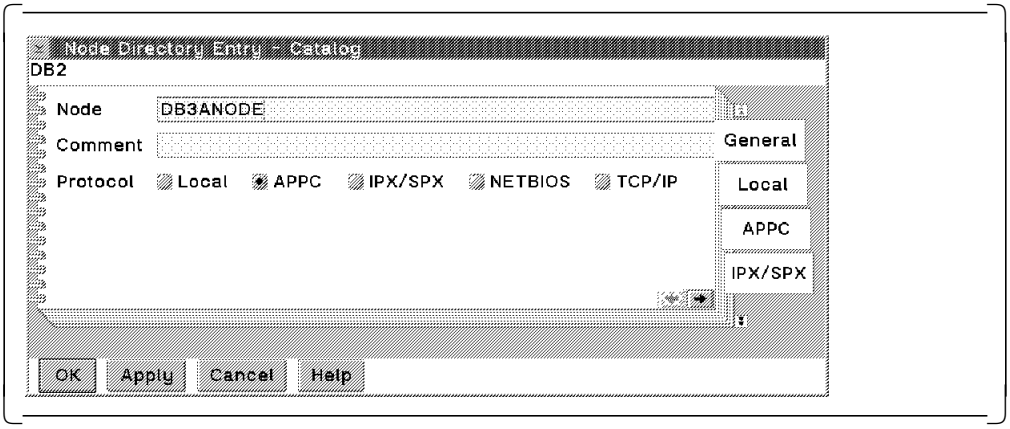


Figure 36. Node Directory Entry - Catalog: General Page

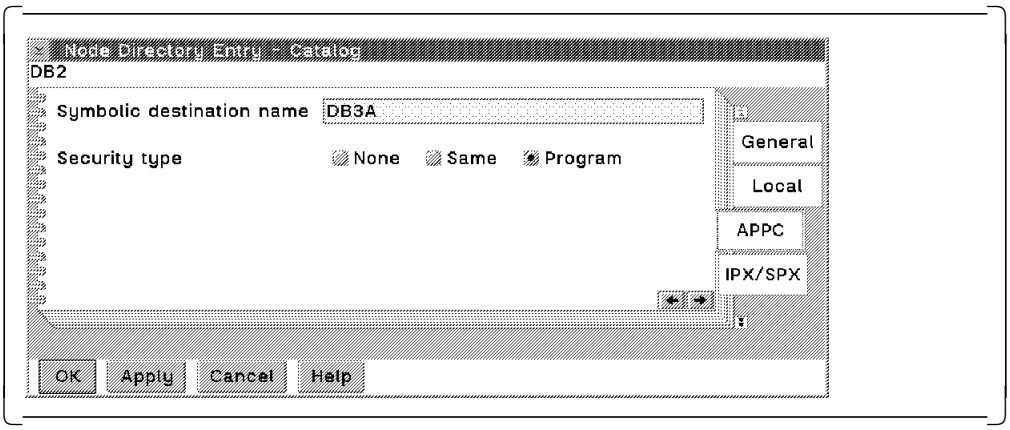


Figure 37. Node Directory Entry - Catalog: APPC Page

Provide the following information:

- Node

  Node is the name by which you reference this node within DDCS.

- Protocol

  When accessing a host system, you use the DRDA protocol, so you select the APPC protocol.

- Symbolic destination name

  The symbolic destination name is the name of the side information that you cataloged in CM/2 or SNA Server/6000 (see Figure 15 on page 21 and Figure 30 on page 37). It defines where the remote node actually resides on the network and which TPN to invoke when the APPC conversation gets there. This value is case sensitive.

- Security

  In order for your transactions to be secure, select Program security. Program security sends both the userid and password to MVS for authentication, where it is checked by RACF.

Cataloging a node from the command line

```
db2 catalog appc node db3anode remote DB3A security program
```

2.6.2 Catalog System Database

The system database directory contains an entry for every database that you can access from this workstation. It stores such information as the node on which the database resides, the name by which the database is known on this workstation, and where security authentication takes place.

In DB2 common server V2.1 you cannot specify the AUTHENTICATION parameter through the database director. If you want to catalog a database with an AUTHENTICATION specification that is different from the instance's AUTHENTICATION, you must use the command line processor:

Cataloging a system database from the command line

```
db2 catalog database db3adcs as db3a at node db3anode authentication dcs
```

- Database name (db3adcs) is, in fact, a pointer to a DCS database directory entry. The DCS database entry describes the name by which the database is known on the MVS system.
- Alias (db3a) is the name by which this database is known within this workstation. It is also the database name that is used in the CONNECT statement.
- Node (db3anode) defines the location of the DRDA AS. It uses the values that were saved in the node directory for the specified node.
- Authentication (dcs) defines where security verification will take place. AUTHENTICATION=DCS specifies that security is implemented by the DRDA AS.

2.6.3 Catalog DCS

The DCS directory stores information about the DRDA AS. For every remote database accessed through DRDA, a DCS database must be cataloged.

To catalog a DCS database, from within the database director perform the following steps:

1. Click on the sign next to the Database Managers icon.
2. Click on the sign next to the DB2 icon.
3. Click on the sign next to the Directories icon.
4. Click on the Database connection services with the right mouse button.
5. Select the Open as details menu item.
6. Select the Directory entry menu.
7. Select Catalog... .

These actions open the window shown in Figure 38 on page 47.

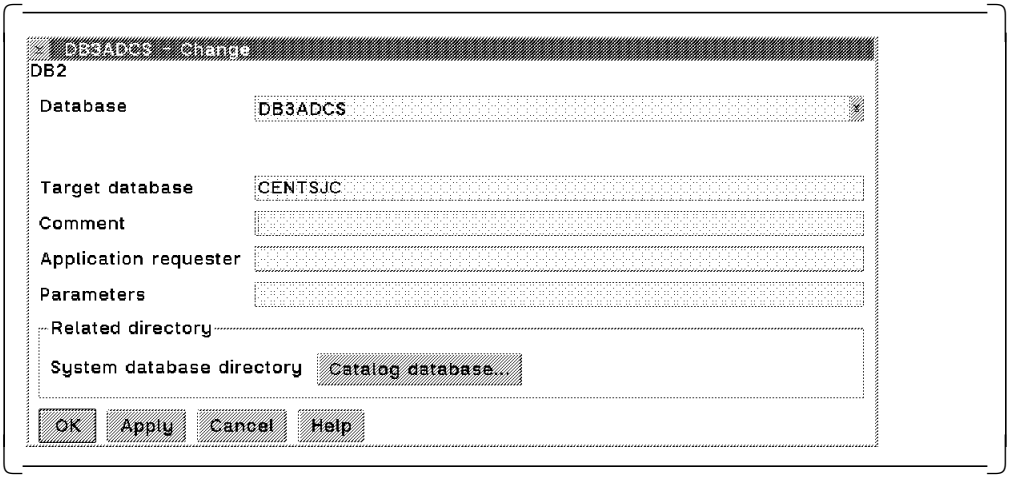


Figure 38. DCS Directory Catalog

Provide the following information:

- Database

  The database name is the name of the DCS database and can be any name of eight characters or less. It is the database parameter that you specify when you catalog a database in the system database directory.

- Target database

  The target database is the location name as registered in the DB2 for MVS/ESA BSDS. Your DB2 for MVS/ESA administrator should be able to tell you what the location name is.

Cataloging a DCS database from the command line

```
db2 catalog dcs database db3adcs as centsjc
```

2.7 Updating Database Manager Configuration for DRDA AS

The database manager configuration must contain several pieces of information in order for your workstation to act as a DRDA AS. In this section we describe the steps involved in using the database director to update the database manager configuration. You must stop and restart DB2 common server after making the updates.

2.7.1 Registering TPN with DB2

When an AR tries to connect to DB2 common server, one of the parameters it passes is the TPN that is to be invoked. This TPN must be defined in both the database manager configuration and the SNA communications product for your workstation (CM/2 or SNA Server/6000).

In OS/2, when DB2 common server is started, it starts a process that listens for incoming connections with that TPN. In AIX, SNA Server/6000 starts this process as part of its own startup process. When the listener process receives an incoming request, it establishes the connection and starts a DB2 agent to handle it.

To specify to DB2 which TPN to listen for, perform the following steps (from within database director):

1. Click on the sign next to the Database Managers icon.
2. Click on the DB2 icon with the right mouse button.
3. Click on the Configure... menu option.
4. Click on the Protocol tab.

The first page of the Protocol tab is displayed (see Figure 39).

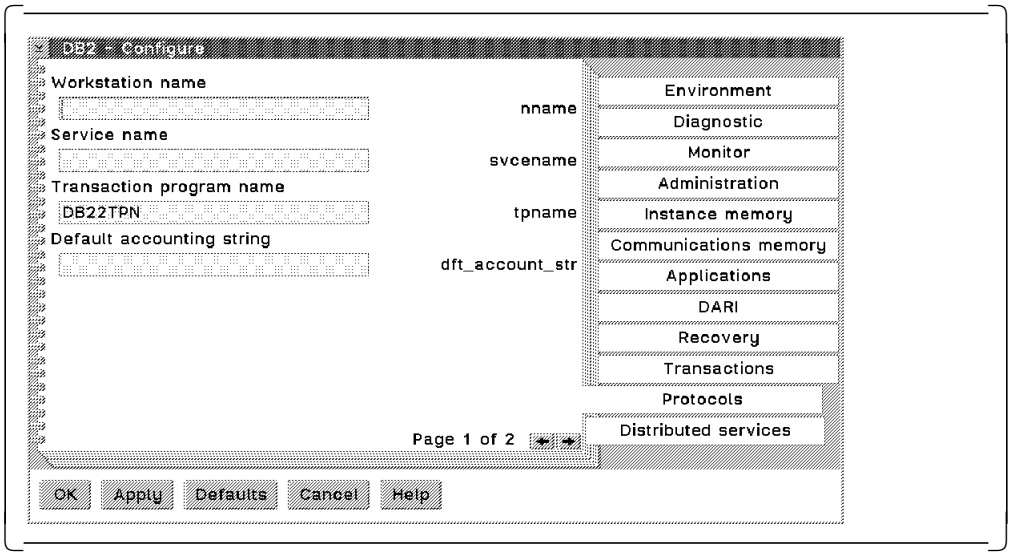


Figure 39. Defining TPN to DB2 common server

The value you enter in the Transaction program name field must be cataloged in your SNA communications product as a TPN (see Figure 19 on page 26 for OS/2 and Figure 34 on page 41 for AIX).

Defining TPN to DB2 common server from the command line

```
db2 update database manager configuration using tpname DB22TPN
```

where DB22TPN is the name of the TPN that you cataloged in your SNA communications product.

2.7.2 Add System Database Entry

When the DRDA AR connects to the DB2 common server system, it passes the alias name of the database that it wants to use. Accordingly, this database alias name must be cataloged in the system database directory.

Note: DB2 for MVS/ESA expects databases in the LOCATION column of SYSIBM.SYSLOCATIONS table to be unique (refer to 2.8.4.1, “CDB Definitions” on page 54). This expectation may force you to catalog an alias for your database. For example, many users have a database called SAMPLE on their workstations, but each user must select a unique name by which to reference his or her SAMPLE database from MVS. Let us assume that we will use OURSAMP as the reference to our real SAMPLE database.

Perform the following steps from within database director:

1. Click on the sign next to the Database Managers icon.
2. Click on the sign next to the DB2 icon.
3. Click on the sign next to the Directories icon.
4. Click on the System Database Directory with the right mouse button.
5. Select the Open as details menu item.
6. Select the Directory entry menu.
7. Select Catalog....

These actions display the System Database Directory Entry - Catalog notebook (see Figure 40, Figure 41, and Figure 42 on page 50).

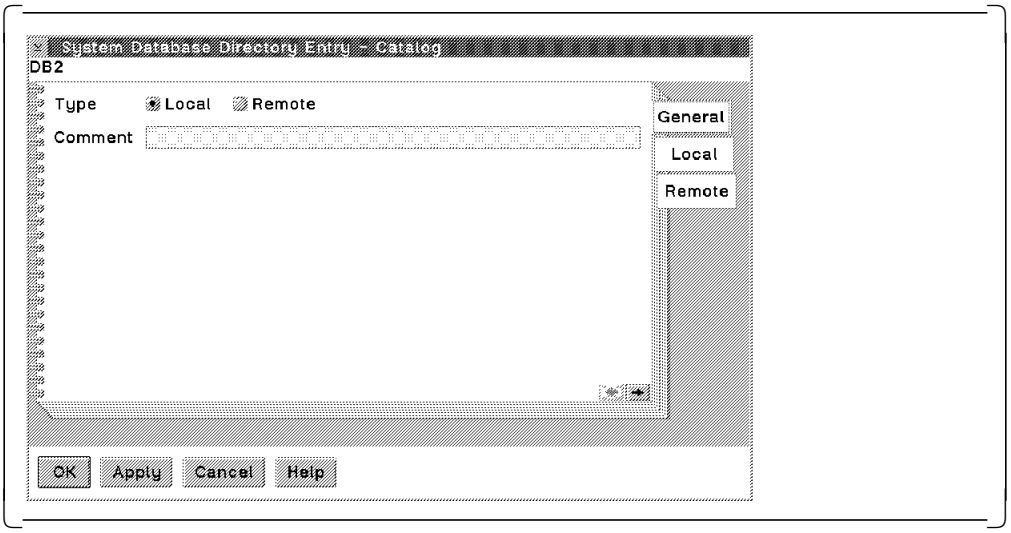


Figure 40. System Database Directory Entry - Catalog: General Page

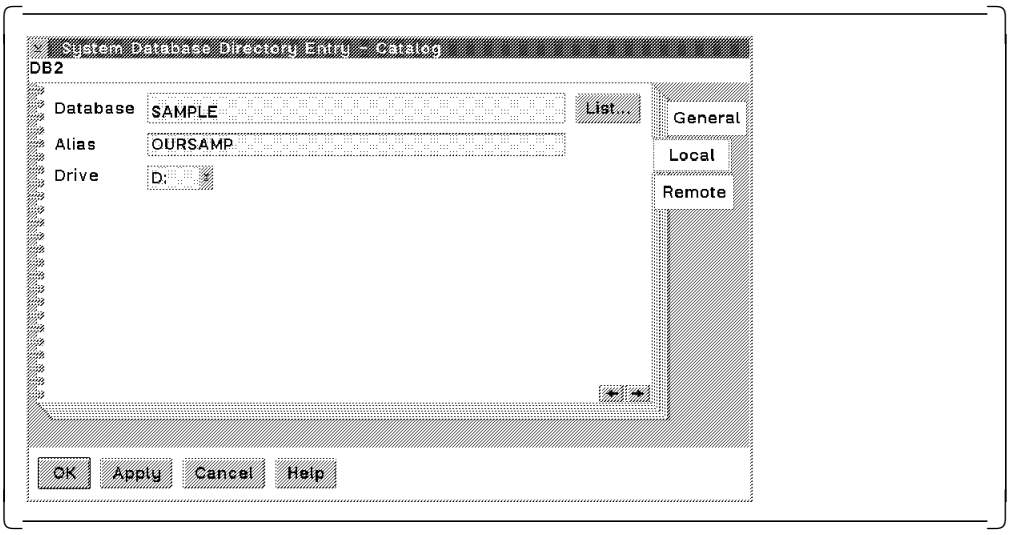


Figure 41. System Database Directory Entry - Catalog: Local Page in OS/2

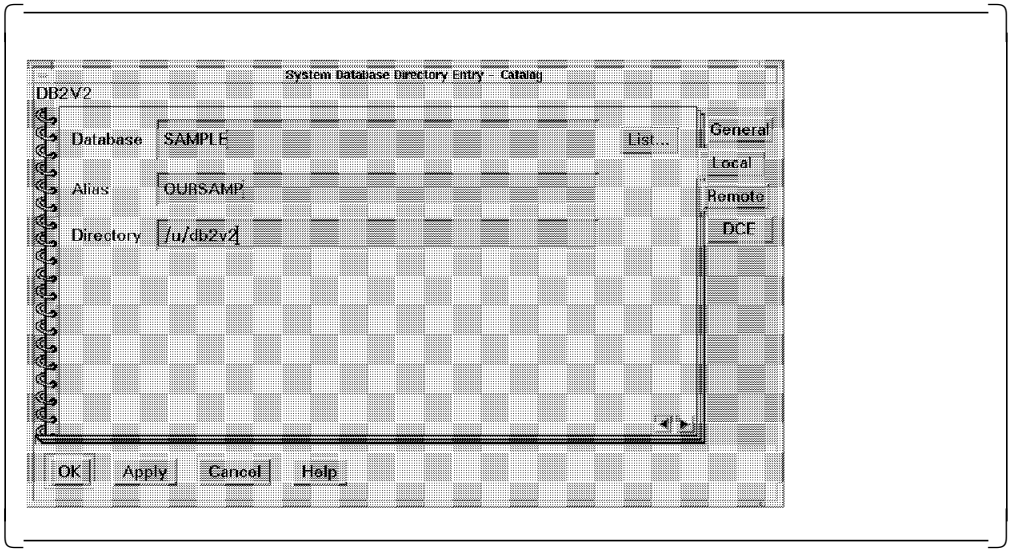


Figure 42. System Database Directory Entry - Catalog: Local Page in AIX

Provide the following information:

- Type

  Type specifies whether the database is located locally or remotely. Because DRDA connections must access local databases in the workstation, select the Local radio button.

- Database

  Database is the name of the physical database that you want to reference.

- Alias

  Alias is the name by which the physical database is referenced. It is also the name that is specified in the LOCATION column in the SYSIBM.SYSLOCATIONS table on MVS (see 2.8.4.2, “AR Requirements” on page 55).

- Drive/Directory

  Drive/Directory (in OS/2 and AIX, respectively) specifies the physical location of the database. In OS/2, it is a drive letter. In AIX, it is a directory name.

Cataloging a system database from the command line

```
db2 catalog database sample as oursamp
```

2.7.3 Set Authentication Location

DB2 common server can perform authentication on either the client or the server. However, for authentication to take place on the server, DB2 common server must be able to extract the userid and password from the SNA communications product for authentication.

In CM/2 V1.11 and previous versions, the password is passed as part of the FMH-5 header, so CM/2 cannot provide the password to DB2 for OS/2. Therefore, you must set authentication client in DB2 for OS/2. This restriction will be removed in a future release of CM/2.

From within database director, perform the following steps:

1. Click on the sign next to the Database Managers icon.
2. Click on the DB2 icon with the right mouse button.
3. Click on the Configure... menu option.
4. Click on the Administration tab.
5. Click on the **->** button to go to page 2.

These actions display the second page of the Administration tab (see Figure 43).

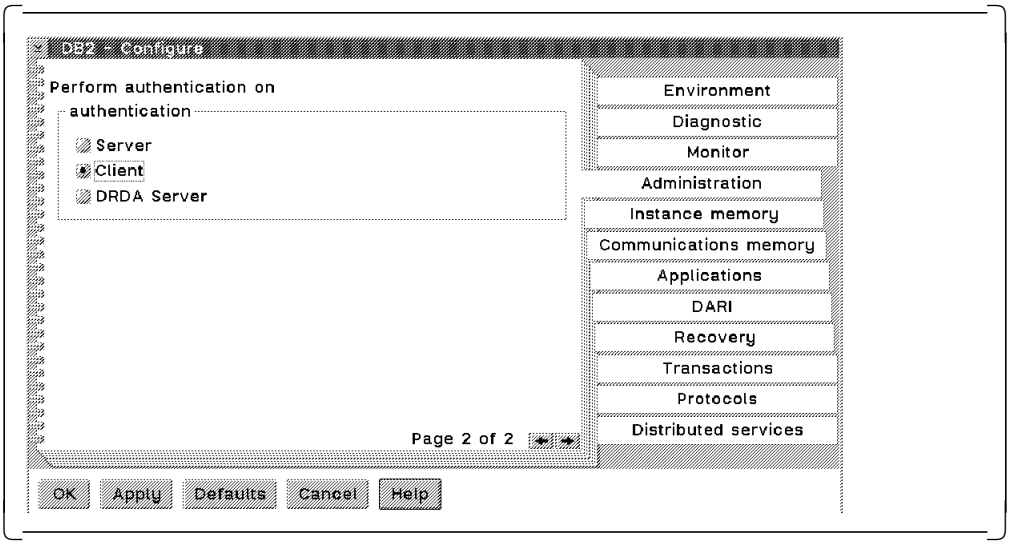


Figure 43. Authentication Client

Specifying authentication location from the command line

```
db2 update database manager configuration using authentication client
```

You must stop and restart DB2 common server to make this change effective.

2.8 DB2 for MVS/ESA in DRDA Environment

In this section we describe what you have to do to make DB2 for MVS/ESA behave as an AR and an AS. We discuss the following topics:

- Network definitions
- Objects for workstation users
- Distributed Data Facility
- Communications database
- Workstation environment
- Running application programs.

If you are an experienced DRDA user on the MVS platform, you can skip to 2.8.4.2, “AR Requirements” on page 55.

2.8.1 Network Definitions

You must define a VTAM application for DB2 for MVS/ESA, as well as define the workstation and mode. These definitions are necessary for both environments (AR and AS).

A VTAM application for DB2 for MVS/ESA must be defined in the VTAM network. Figure 44 shows a sample APPL definition. SCLUDB3A maps to the LU name.

```
*-------------------------------------------------------------------*
*      APPL STATEMENT FOR SYSTEM DB3A                               *
*-------------------------------------------------------------------*
SCLUDB3A APPL ACBNAME=SCLUDB3A, NETID  FOR DB2                       X
              APPC=YES,         VTAM APPLICATION                     X
              ATNLOSS=ALL,                                           X
              AUTH=(ACQ),       ACCESS TO VTAM FUNCTIONS             X
              AUTOSES=10,                                            X
              DMINWNL=25,                                            X
              DMINWNR=25,                                            X
              DSESLIM=50,       MAX NUMBER OF SESSIONS IN USE        X
              EAS=509,                                               X
              ENCR=NONE,                                             X
              MODETAB=AGWTAB,   MODETAB NAME                         X
              PARSESS=YES,                                           X
              SECACPT=ALREADYV, PASSWORD VERIFICATION ALREADY VERIFIED X
              SONSCIP=NO,                                            X
              SRBEXIT=YES,                                           X
              SYNCLVL=SYNCPT,                                        X
              VERIFY=NONE,                                           X
              VPACING=2,        PACING PARAMETER                     X
              VTAMFRR=NO
```

Figure 44. DB2 as VTAM Application

The LU name is also registered as the LUNAME field in the communication record of the BSDS (see Figure 45). Use the change log inventory utility, DSNJU003, to update this BSDS information (see the *DATABASE 2 Command and Utility Reference*).

```
                **** DISTRIBUTED DATA FACILITY ****
                       COMMUNICATION RECORD
                     11:45:11 JULY 26, 1995
LOCATION=CENTSJC  LUNAME=SCLUDB3A  PASSWORD=(NULL)
```

Figure 45. Communication Record

The workstation must be defined to VTAM in a PU macro. An independent LU (LOCADDR=0) or dependent LU (LOCADDR not=0) should be associated with the workstation. Dependent LUs can be used only when the workstation functions as an AR. Figure 46 on page 53 shows a sample of a PU and an independent LU for the workstation.

```
*-------------------------------------------------------------------*
*      DB2 DEFAULT MODENAME FOR APPLICATION-DIRECTED ACCESS         *
*-------------------------------------------------------------------*
*
SJA2035  PU    ADDR=01,                                          X
               IDBLK=05D,                                        X
               IDNUM=A2035,                                      X
               ANS=CONT,                                         X
               DISCNT=NO,                                        X
               IRETRY=NO,                                        X
               ISTATUS=ACTIVE,                                   X
               MAXDATA=265,                                      X
               MAXOUT=7,                                         X
               MAXPATH=1,                                        X
               PUTYPE=2,                                         X
               SECNET=NO,                                        X
               MODETAB=POKMODE,                                  X
               DLOGMOD=DYNRMT,                                   X
               USSTAB=USSRDYN,                                   X
               PACING=1,
               VPACING=2
*
SJA2035A LU    LOCADDR=002,LOGAPPL=SCGVAMP
SJA2035B LU    LOCADDR=003,LOGAPPL=SCGVAMP
SJA2035I LU    LOCADDR=0,DLOGMOD=LU62APPB
SJA2035J LU    LOCADDR=0,DLOGMOD=LU62APPB
SJA2035K LU    LOCADDR=0,DLOGMOD=LU62APPA
SJA2035L LU    LOCADDR=0,DLOGMOD=LU62APPA
*
```

Figure 46. PU and Independent LU for Workstation

You can also associate the LU and the PU with the ALSLIST parameter of the CDRSC macro, using cross-domain resources:

```
lu-name CDRSC ALSLIST=(pu-name)
```

The DB2 AR uses two default mode names: IBMDB2LM for private protocol, and IBMRDB for application-directed (DRDA). Both modes should be included in the mode table available to DB2 for MVS/ESA. Figure 47 shows an example of IBMRDB.

```
*-------------------------------------------------------------------*
*      DB2 DEFAULT MODENAME FOR APPLICATION-DIRECTED ACCESS         *
*-------------------------------------------------------------------*
IBMRDB  MODEENT  LOGMODE=IBMRDB,   LOGMODE NAME                    X
                 TYPE=0,                                           X
                 PSNDPAC=X'00',                                    X
                 SSNDPAC=X'02',                                    X
                 SRCVPAC=X'00',                                    X
                 RUSIZES=X'8989',                                  X
                 FMPROF=X'13',                                     X
                 TSPROF=X'07',                                     X
                 PRIPROT=X'B0',                                    X
                 SECPROT=X'B0',                                    X
                 COMPROT=X'50A5',                                  X
                 PRIPROT=X'B0',                                    X
                 PSERVIC=X'060200000000000000122F00'   LU6.2 TYPE
```

Figure 47. IBMRDB Mode Definition

2.8.2 Objects for Workstation Users

In this section we highlight the differences between some objects in a workstation and in an MVS environment. If you are familiar with these differences you can skip this section.

2.8.2.1 DB2 for MVS/ESA Subsystem

A DB2 for MVS/ESA subsystem (in a non-data sharing environment) has some similarities with a DB2 common server database. Both have their own catalog and logs. In terms of connections, in the workstation environment you connect to a database; in the MVS environment you connect to a location (or RDBname) that identifies a DB2 subsystem. In summary, a DB2 for MVS/ESA subsystem consists of one catalog, one set of logs, and multiple databases.

2.8.2.2 Database in the DB2 for MVS/ESA Environment

The concept of a database in the DB2 for MVS/ESA environment is different from the concept of a database in a workstation environment. A database in the DB2 for MVS/ESA environment can be regarded as a set of objects, such as user tables and indexes. In a way, it is an administrative facility; you can, for example, issue a command to start or stop the entire database. Preventing access to a DB2 for MVS/ESA database does not prevent connection to the DB2 for MVS/ESA subsystem nor does it prevent access to other accessible databases.

2.8.3 Distributed Data Facility

Distributed data facility (DDF) is the LU 6.2 application program for DB2 for MVS/ESA. It must be started to support any kind of communication (requester or server). DDF uses information stored in the communication database to establish connections and send data.

2.8.4 Communication Database

In this section we briefly introduce the communication database (CDB). We cover the following topics:

- CDB definitions
- AR requirements
- AS requirements
- Enabling CDB updates.

For complete information about the CDB refer to the *DATABASE 2 Administration Guide*.

2.8.4.1 CDB Definitions

The CDB contains such information as locations, LU names, and modes. The CDB has five tables, all prefixed by SYSIBM:

- The **SYSLUNAMES** table defines the remote system's LU names, security, and mode requirements for conversations.
- The **SYSLOCATIONS** table defines the network location name (or RDBname), LU name, and TPN. The location name must be unique. It is used by the DB2 for MVS/ESA AR.
- The **SYSLUMODES** table defines for a specific LU name the mode name (could be generic or a specific LU name), the maximum number of

concurrently active conversations when CNOS processing occurs, and whether sessions should be preallocated.

- The **SYSMODESELECT** table defines the mode name for a specific authorization-id, plan name, and LU name.
- The **SYSUSERNAMES** table defines the outbound and/or inbound translation for a LU name and/or authorization-id.

The tables have referential constraints as shown in Figure 48.

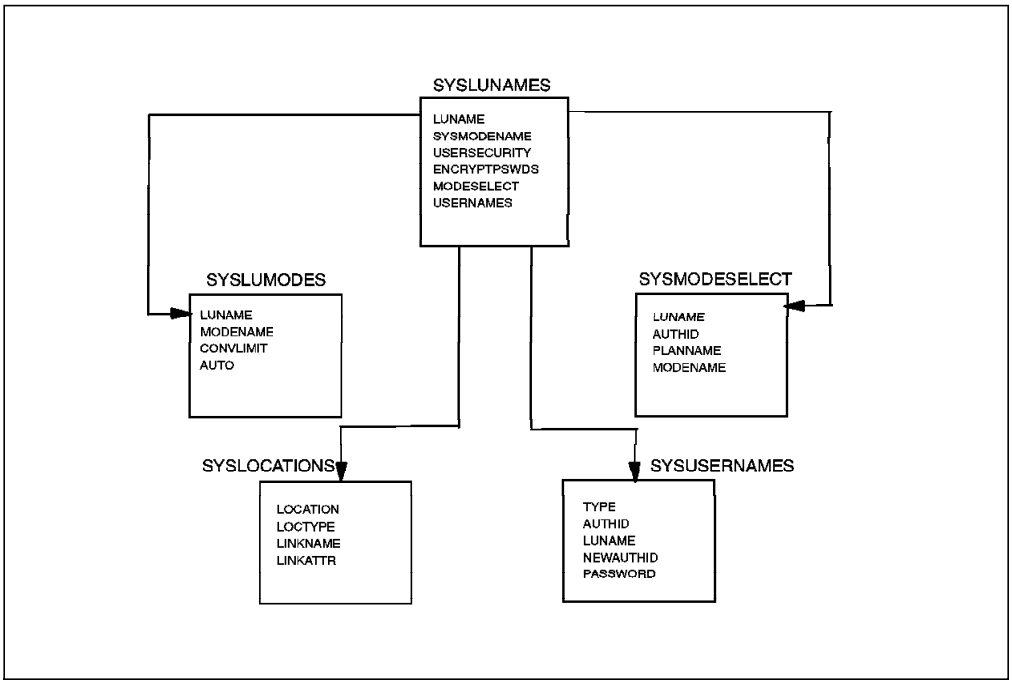


Figure 48. Referential Constraints in the CDB

2.8.4.2 AR Requirements

The trip from DB2 for MVS/ESA to DB2 common server (DB2 for OS/2, DB2 for AIX) begins with the CONNECT statement. The location name (or RDBname) is passed in the CONNECT statement. The location must be defined in the SYSLOCATIONS table. Below we explain the definitions required for the AR.

SYSLOCATIONS: An entry is defined in the SYSLOCATIONS table for each server. The LINKATTR value to be used must match the TPNAME parameter of the DB2 common server database manager configuration file and the network definition of the DB2 common server partner environment.

The LOCATION value in SYSLOCATIONS must match the database alias name defined in the system database directory of the workstation (see Figure 49 on page 56).

If you have to connect to existing databases in the workstation environment and those databases have the same name, you can issue the DB2 CATALOG DATABASE command to differentiate them.

To insert a row in SYSIBM.SYSLOCATIONS use the following SQL statement:

```
INSERT INTO SYSIBM.SYSLOCATIONS (LOCATION,LINKNAME,LINKATTR)
VALUES('SAMPLEP','SNC4862','DB2JTPN')
```

Remember that there are referential integrity constraints among the CDB tables, so you should first insert a corresponding row in the SYSLUNAMES table.

SYSLUNAMES: A description of the LU name should be placed in SYSLUNAMES. Information such as whether to use the default mode (IBMRDB for application-directed access), or search within SYSMODESELECT for the appropriate mode name, outbound translation, and user security (already verified or conversational) can also be specified.

To insert a row in SYSLUNAMES use the following SQL statement:

```
INSERT INTO SYSIBM.SYSLUNAMES
(LUNAME,USERSECURITY,USERNAMES) VALUES('SNC4862','A','O')
```

SYSUSERNAMES: If outbound translation is required, a translated userid and an accompanying password must be sent to the server. SYSUSERNAMES contains this information.

To insert a row in SYSUSERNAMES use the following SQL statement:

```
INSERT INTO SYSIBM.SYSUSERNAMES
VALUES('O','','SNC4862','USERID','PASSWORD')
```

As shown in Figure 49, the communication information—partner LU (LU name), TPN (TPNAME), and mode—required to establish the connection is read by DDF from CDB tables. DDF issues the request for the connection between the two LUs to VTAM.

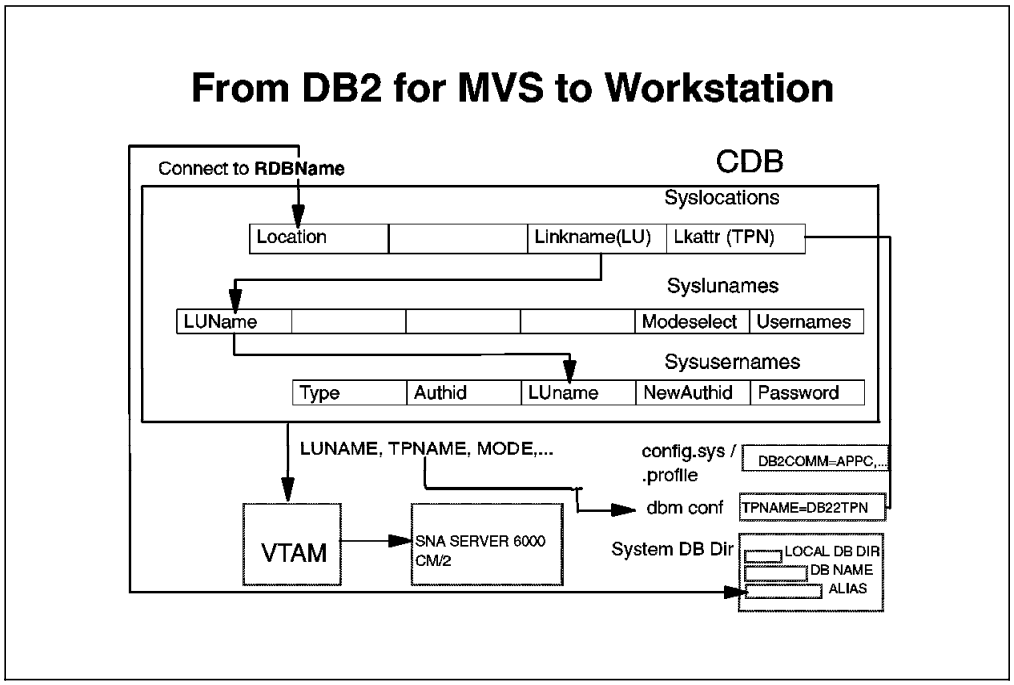


Figure 49. DB2 for MVS/ESA AR: Workstation AS

Notes:

1. To insert rows in the CDB, use SPUFI on the MVS platform or the command line processor (CLP) from the workstation platform.
2. You do not have to have a row for the local DB2 for MVS/ESA in the CDB.

2.8.4.3 AS Requirements

If you insert a blank row in SYSLUNAMES, DB2 for MVS/ESA accepts requests from any AR using the default DB2 for MVS/ESA security values. Update the CDB only if the defaults do not apply.

2.8.4.4 Enabling CDB Updates

Any table in the CDB can be updated while DDF is active. The changes take effect as follows:

- Changes to SYSLUNAMES, SYSLUMODES, and SYSLOCATIONS take effect the next time DDF is started.
- Changes to SYSUSERNAMES and SYSMODESELECT take effect at the next thread access.

In all cases, existing conversations continue to operate as the table specified before the update.

Note: The process of modifying or dropping could return a negative SQLCODE. When DDF is active, DB2 commands in the CDB are not allowed.

2.8.5 Workstation Environment

Below we summarize the activities you must perform on the workstation to have it act as an AS.

2.8.5.1 DB2 common server

- In order for DB2 common server to accept APPX clients, you must set the DB2COMM environment variable to APPC in the CONFIG.SYS file for OS/2 and .profile for AIX.
- The LINKATTR value specified in DB2 for MVS/ESA (from SYSLOCATIONS) must match the TPNAME parameter of the database manager configuration.
- The location (or RDBname) in the CONNECT statement must match the alias name of an entry in the system database directory of DB2 common server.
- Depending on the security requirements, a partner LU relating to the LU name of DB2 for MVS/ESA must be defined. Security is discussed in Chapter 4, “Security Considerations” on page 79.
- The application program must be bound to the DB2 common server.

For a description of how to update DB2 common server see 2.3, “Configuring DB2 for OS/2 As a DRDA AS” on page 25 for DB2 for OS/2 and 2.5, “Configuring DB2 for AIX As a DRDA AS” on page 40 for DB2 for AIX.

2.8.5.2 OS/2

- There must be matching CM/2 definitions for the local node. The local node ID must match IDBLK and IDNUM. The local node name must match the LU within the CDB and the LU with LOCADDRR=0 for the PU representing OS/2.
- The mode name and TPN name specified by DB2 must be defined in CM/2.
- A link from the workstation to VTAM must be active.

For a description of how to update database directories see 2.2, “Configuring DDCS for OS/2 As a DRDA AR” on page 15.

2.8.5.3 AIX

- The LINKNAME column of SYSLOCATION specifies the partner LU that represents DB2 for AIX. This partner must match an LU (independent) defined in VTAM for the AIX machine (LOCADDR=0) and be defined in a local LU profile of the SNA Server/6000.
- The LINKATTR column indicates with which TPN DB2 for MVS/ESA should try to establish a conversation when going to DB2 for AIX. A corresponding TPN profile must be configured in the SNA Server/6000, and the profile name must match the TPNAME parameter of the database manager configuration file of this DB2 for AIX instance.
- The mode name used by DB2 for MVS/ESA must be configured in the SNA Server/6000.
- The SNA Server/6000 control point profile, the Token Ring Link station profile, and the Token Ring SNA DLC profile must be configured to allow DB2 for MVS/ESA to connect to DB2 for AIX.

For a description of how to update database directories see 2.4, “Configuring DDCS for AIX As a DRDA AR” on page 30.

2.8.6 Running Applications

All programs that you run in a DRDA client/server environment must be prepared in a remote package that contains static or dynamic SQL statements and bound into the DB2 common server. See Chapter 3, “Binding” on page 59 for more information about binding.

Chapter 3. Binding

In this section we describe the DB2 bind process parameters that you can specify, explain how to bind some DB2 for MVS/ESA sample programs to the DB2 common server AS, and review some considerations when binding the DB2 common server utilities to a DB2 for MVS/ESA AS. We discuss the following topics:

- Introduction to the bind process
- Main BIND parameter differences in a DRDA environment
- Other BIND/PREP parameters
- Binding DB2 for MVS/ESA sample programs
- Binding DB2 common server utilities.

If you are familiar with the bind process in a DRDA environment, skip to 3.2, “Cross-Platform DRDA Bind Differences” on page 62.

3.1 Introduction to the Bind Process

In this section we describe the bind process for DB2 for MVS/ESA and DB2 common server in a DRDA environment.

To execute a program in a DRDA environment, follow these steps:

Precompile This process modifies the source application program converting the SQL statements into DB2 run-time API calls, and create files that are input to the bind process.

Compile Compile is the process that converts source application programs into object modules without SQL statements.

Link-edit This process converts object modules into load (or executable) modules.

Bind This process creates a package at the AS.

3.1.1 DB2 for MVS/ESA Environment

Any application executing in a DB2 for MVS/ESA AR environment must have a plan associated with it at execution time. A plan is a list of packages available for the application program. Packages can be local (for local execution) or remote (for remote execution). You must bind the plan indicating the list of local and remote packages.

Below we describe the process of creating packages and plans. Refer to Figure 50 on page 61.

You invoke the precompiler (DSNHPC) that generates a database request module (DBRM) and a modified source. The modified source is input to the compiler. The compiler passes the object module to be link-edited. The link-edit process generates the executable load module.

If the application program connects to a remote database, the DBRM must be bound into a package in the respective AS, and the plan associated with the program must include the package name.

Use the BIND PACKAGE command to bind one DBRM in a local or remote package. Here is an example of the BIND PACKAGE command.

```
BIND PACKAGE (location.collection-id) MEMBER(DBRM1) ACTION(REPLACE)
```

For this example, if the package is bound to a DB2 common server AS, the *location* is the remote database alias name, *collection-id* is mapped to the creator of the plan, and *DBRM1* identifies the DBRM and the package name.

To issue the BIND PACKAGE command for a remote package you must have one of the following authorities in the DB2 common server AS:

- SYADM or DBADM authority
- BINDADD authority
- BIND privilege on the package, if it exists.

Note: You do not need any authority to issue a BIND PACKAGE command from the AR.

The BIND PLAN command is used to create the DB2 for MVS/ESA plan. For DRDA you specify a package list as input to the BIND PLAN command. Here is an example of the BIND PLAN command:

```
BIND PLAN(plan name) PKLIST(location.collection-id.packagename) ACTION
(ADD)
```

Note that you can specify for *location*, *collection-id*, or *packagename* the wildcard character, *, indicating all occurrences of locations, collections, or packages.

To run the BIND PLAN command you must have one of the following authorities in the DB2 for MVS/ESA AS:

- SYADM or SYSCTRL authority
- BINDADD authority
- BIND privilege on the package, if it exists
- BINDAGENT privilege granted for the package's owner, if it exists
- PACKADM privilege on the collection-id, if it exists.

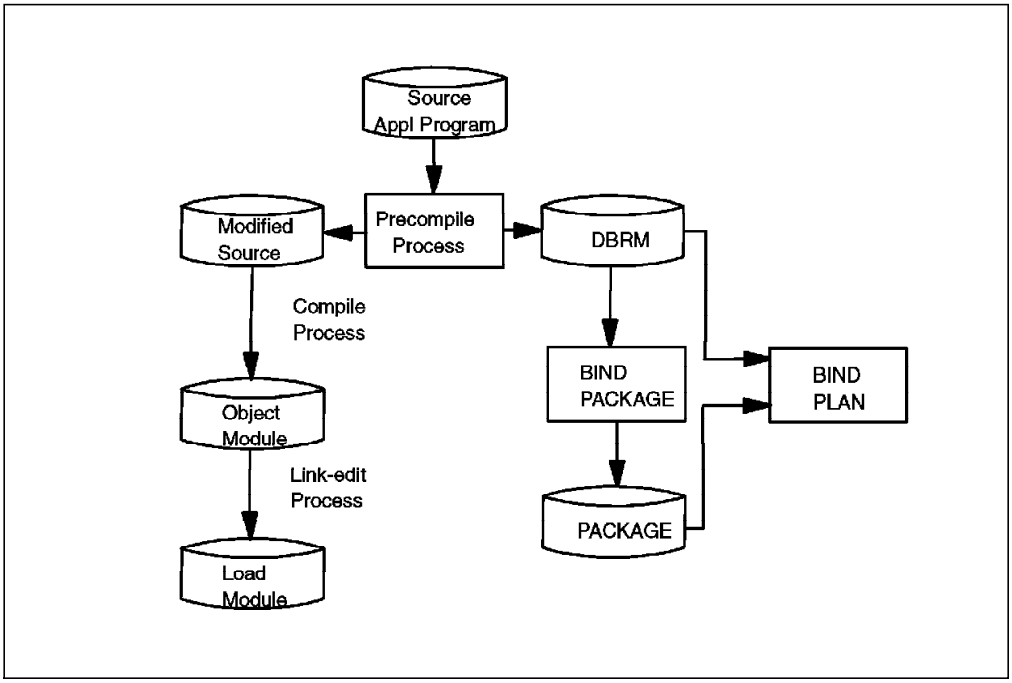


Figure 50. Creating Packages and Plans in the MVS Environment

You can bind packages and plans using the interactive facilities of DB2 for MVS/ESA (DB2I), or by submitting a batch job.

Note: See *DATABASE 2 Application Programming and SQL Guide* for more detailed information.

3.1.2 DB2 common server Environment

To create a remote package from the DB2 common server environment in the AS you use the PRECOMPILE (PREP) or BIND command. See *DATABASE 2 Application Programming Guide for common server Version 2* for detailed information on these commands. You must first be connected to the remote location before issuing the PREP or BIND command. Refer to Figure 51 on page 62.

The PREP command generates a modified source program and, optionally, the remote package:

```
DB2 PREP source_application_program BINDFILE PACKAGE
```

When you use the PACKAGE option, a package is created at the AS. When you use the BINDFILE option, a bind file is created that contains the data required to create a package. The default of the bind file is application_name and extension "BND."

If you specify a BINDFILE and do not specify the PACKAGE option, a package is not created. In this case you must issue the BIND command to create the package. Here is an example of the BIND command:

```
DB2 BIND application_name.BND
```

To run the BIND or PREP command to create a remote package on a DB2 for MVS/ESA AS you must have one of the following authorities in the DB2 for MVS/ESA AS:

- SYADM or SYSCTRL authority

- BINDADD authority
- BIND privilege on the plan, if it exists
- BINDAGENT privilege granted for the plan's owner, if it exists.

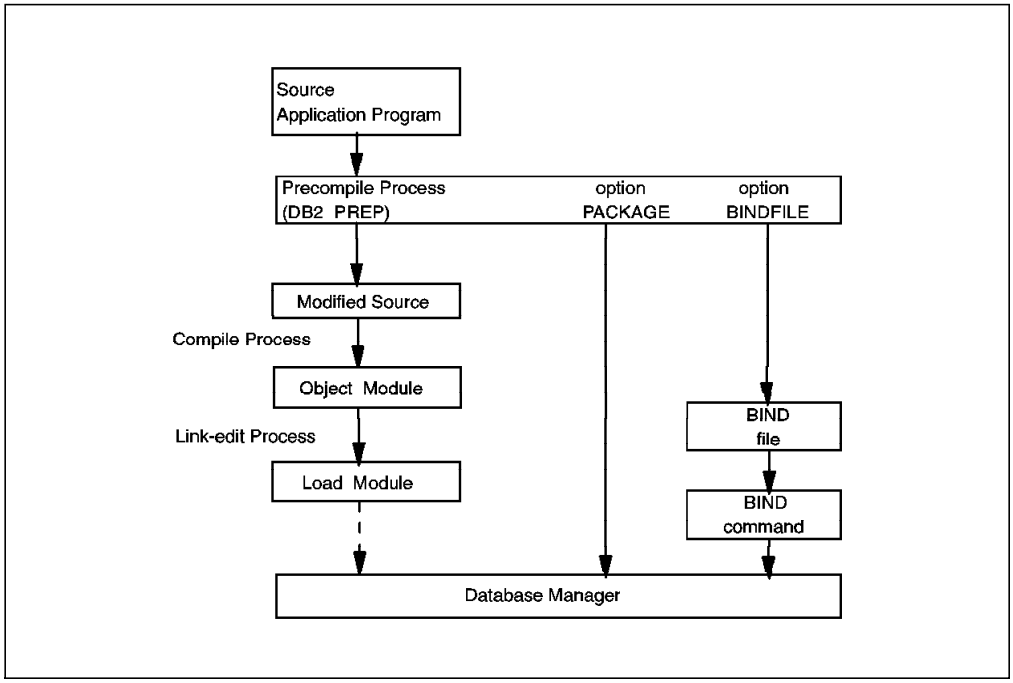


Figure 51. Package Creation in DB2 common server

3.2 Cross-Platform DRDA Bind Differences

In this section we discuss the following topics:

- BIND ID resolution
- Isolation levels
- Blocking
- Checking SQL for DB2 for MVS/ESA.

3.2.1 Bind ID Resolution

In this section we describe the ID resolution during the bind process in DB2 common server and DB2 for MVS/ESA.

3.2.1.1 Remote Bind to DB2 common server

DB2 for MVS/ESA sends out several IDs associated with the BIND request. Refer to Figure 52 on page 63. These IDs are specified in the BIND PACKAGE command and are also taken from the binder. For the following DB2 for MVS/ESA BIND command:

```
BIND PACKAGE(location.collection-id) OWNER(USER2) QUALIFIER(USER2)
```

- The OWNER is an authorization-id that requires privileges for all static SQL within the program. If not specified, it defaults to the binder.

- The QUALIFIER is used as the authorization-id for unqualified names of tables and/or views within the program. If not specified, it defaults to the OWNER.

DB2 common server does not support QUALIFIER or OWNER parameters, unless those parameters have the same value of the binder (primary-id), or, if you have outbound translation, the same value as the NEWAUTHID column in the SYSIBM.SYSUSERNAMES table. If the parameter values do not match, you get an SQLCODE -4930 (invalid option).

The binder ID is recorded in the BOUNDBY column of SYSIBM.SYSPLAN of the DB2 common server. It must have the privilege to execute all static SQL within the program (or have DBADM or SYSADM), and the collection-id is recorded in the CREATOR column of SYSIBM.SYSPLAN of the DB2 common server AS.

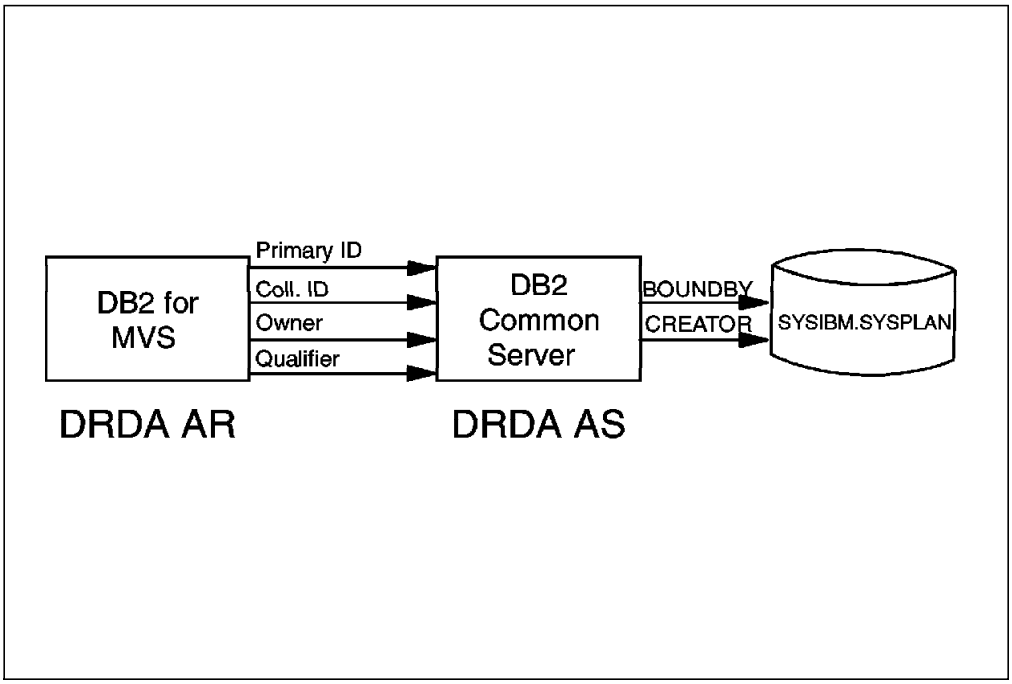


Figure 52. Remote Bind to DB2 common server

The binder authorization-id is replaced by the translated authorization-id if you specify outbound translation in DB2 for MVS/ESA. The OWNER and QUALIFIER parameters do not translate with outbound translation. To summarize the binder, owner, and qualifier must match the primary authorization-id or the outbound translated authorization-id, but the collection may be different.

3.2.1.2 Remote Bind to DB2 for MVS/ESA

DB2 common server sends out several IDs associated with the BIND request. The IDs are specified in the BIND or PREP commands and are also taken from the binder. Refer to Figure 53 on page 64.

OWNER and QUALIFIER are new PREP and BIND DRDA parameters. They are not supported locally by DB2 common server, but you can specify them when preparing or binding remote packages to other DRDA ASs, such as DB2 for MVS/ESA.

The OWNER parameter specifies the authorization-id for which privileges are checked for all of the static SQL within the program. It can be a different

authorization-id from the authorization-id of the binder (primary-id). The default for owner is the primary-id of the bind process.

The QUALIFIER is used as the authorization-id for unqualified names of tables or views within the program. The default for qualifier is the owner′s authorization-id, whether OWNER is explicitly specified or defaulted.

The COLLECTION parameter, which is a new parameter, specifies the DB2 for MVS/ESA collection name of the package. If COLLECTION is not specified, it defaults to the authorization-id of the user processing the package.

The collection-id is also referred to as a schema name in DB2 common server. Using the BIND command you can bind a package into a collection different from the collection specified in the PREP command, although the collection specified in the PREP command will be hard-coded in the executable file.

DB2 common server has a new SQL statement, “SET CURRENT PACKAGESET command collection-id,” to change the collection-id at run-time. This statement allows an application program to specify the collection-id used when selecting a package for an executable SQL statement. You can embed this statement only in an application program. It is not executable in dynamically prepared statements and in REXX.

Here is an example of the PREP command specifying COLLECTION, OWNER, and QUALIFIER.

```
DB2 PREP application_program_name COLLECTION(COLL1)
BINDFILE OWNER(USER1) QUALIFIER(USER2)
```

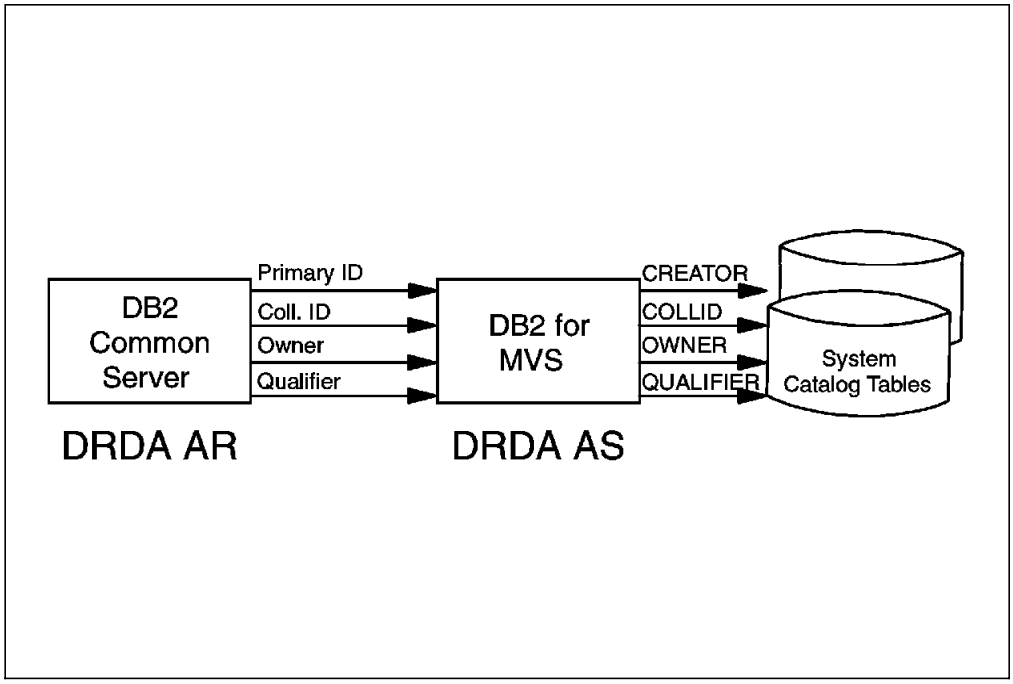


Figure 53. Remote Bind to DB2 for MVS/ESA

To summarize, when binding from DB2 common server to DB2 for MVS/ESA, you can have different values for collection, owner, and qualifier parameters. Neither of them can match the primary authorization-id.

3.2.2 DRDA Isolation Levels

The isolation level associated with an application process is the degree of isolation of that process from other concurrently executing application processes. The isolation level of an application process specifies:

- The degree to which rows read and updated by the process are available to other processes
- The degree to which update activity of other concurrently executing processes can affect your process.

Regardless of the isolation level, the DBMS places exclusive locks on every row that is updated, inserted, or deleted. Thus, all isolation levels ensure that any row that is changed during a unit of work is not changed by other processes until the unit of work is completed.

3.2.2.1 Types

The available isolation levels are:

Repeatable read (RR) Ensures that any row read during a unit of work is not changed by other processes until the unit of work is complete. In addition, any row changed by another application process cannot be read until it is committed by that application process. The application process is completely isolated from the effects of concurrent application processes.

Read stability (RS) Ensures that any row read during a unit of work is not changed by other processes until the unit of work is complete. In addition, any row changed by another application process cannot be read until it is committed by that application process. Unlike RR, RS does not completely isolate the application process from other concurrent application processes. With RS you can get different results for the same query as in the following scenario:

1. Application process P1 reads a set of rows.
2. Application process P2 inserts rows that satisfy P1 search condition and commits.
3. P1 reads the set of rows again and obtains both the original rows and the rows inserted by P2.

If RR was specified for application process P1, application process P2 would have to wait for application process P1 to commit in order to insert the rows.

Cursor stability (CS) Like RR, ensures that any row that was changed by another application process cannot be read until it is committed by that application process. Unlike RR, CS only ensures that the current row of every cursor is not changed by other application processes. Thus, the rows that were read during a unit of work can be changed by other application processes.

Uncommitted read (UR) For a SELECT INTO, FETCH with a cursor declared for read only, or a subselect used in a subquery, level UR avoids acquiring locks and allows any row that is read during this unit of work to be changed by other application processes. Any row that was changed by another application process can be read by this process even if the change has not been committed by that application process.

No commit (NC) Specifies that commitment control is not to be used. Neither DB2 common server nor DB2 for MVS/ESA supports NC. You can specify this option, but you receive a SQLCODE +595 indicating the isolation level specified is not supported by DB2 and has been escalated to a supported isolation level. COMMIT and ROLLBACK statements are not required for changes to become visible to concurrent users.

3.2.2.2 Options

DB2 common server supports four isolation levels: RR, RS, CS, and UR even though you can specify NC for use in packages bound to ASs (see Table 2 and Table 4).

DB2 for MVS/ESA V3 supports CS and RR (see Table 4). You cannot specify any other option when binding from DB2 for MVS/ESA V3. DB2 for MVS/ESA V3 does not support row level locking.

DB2 for MVS/ESA V4 supports RR, CS, and UR, although you can specify RS or NC for use in packages bound to ASs other than DB2 for MVS/ESA (see Table 3).

You can specify any level when binding from DB2 for MVS/ESA V4 to DB2 common server, or from DB2 common server to DB2 for MVS/ESA V4; the options not supported are upgraded to the next, more restrictive level.

Table 2. DB2 common server to DB2 for MVS/ESA V4

DB2 common server	**DB2 for MVS/ESA V4**
CS	CS
RR	RR
UR	UR
RS	RR
NC	UR

Table 3. DB2 for MVS/ESA V4 to the DB2 common server

DB2 for MVS/ESA V4	**DB2 common server**
CS	CS
RR	RR
UR	UR
RS	RS
NC	UR

Table 4. Conversion between the DB2 common server and DB2 for MVS/ESA V3

DB2 common server	**DB2 for MVS/ESA V3**
CS	CS
RR	RR
UR	CS
RS	RR
NC	CS

3.2.3 Blocking

Blocking reduces the amount of interactions between the AR and the AS by transmitting a block of rows in a single operation.

For DB2 for MVS/ESA you specify the blocking options through the CURRENTDATA parameter. The options for the CURRENTDATA parameter are YES or NO.

For DB2 common server you specify the blocking options through the BLOCKING parameter. The options for the BLOCKING parameter are UNAMBIG, ALL, or NO.

Refer to Table 5 for the relationship between these two parameters.

The specification of BLOCKING UNAMBIG is equivalent to the specification of CURRENTDATA(YES). Blocking occurs if all of the following conditions are met:

- Fetch-only cursors
- Cursors not specified as FOR UPDATE OF
- Cursors do not have static DELETE WHERE CURRENT OF statements
- Cursors do not have dynamic SQL statements.

UNAMBIGUOUS is the default for DB2 common server. Ambiguous cursors are treated as updatable.

The specification of BLOCKING ALL is equivalent to the specification of CURRENTDATA(NO). Blocking occurs if all of the following conditions are met:

- Fetch-only cursors
- Cursors not specified as FOR UPDATE OF
- Cursors for which no static DELETE WHERE CURRENT OF statements are executed.

CURRENTDATA(NO) is the default for DB2 for MVS/ESA, so BLOCKING ALL is what you get if you do not specify this parameter when binding from DB2 for MVS/ESA.

There is no equivalent to the BLOCKING NO option in the DB2 for MVS/ESA platform. The NO option indicates forced, fixed row, that is, do not block. Although this option is not supported when binding a package from DB2 for MVS/ESA, it is supported for packages that were bound remotely. The DEFERPREP column of SYSIBM.SYSPACKAGE contains a value of A, indicating that blocking is not being requested.

Table 5. Blocking Relationship

DB2 common server BLOCKING	**DB2 for MVS/ESA CURRENTDATA**
UNAMBIG	YES
ALL	NO
NO	---

3.2.4 Checking SQL Syntax for DB2 for MVS/ESA

If you are precompiling an application on the DB2 common server that executes SQL statements on the DB2 for MVS/ESA, consider using the flagger facility to check the syntax of the SQL statements. The flagger indicates SQL syntax that is supported by DB2 common server but not supported by DB2 for MVS/ESA.

You can use the SQLFLAG option on the PREP command to invoke the flagger and specify which version of DB2 for MVS/ESA SQL syntax is to be used for comparison. The following example checks SQL syntax for DB2 for MVS/ESA V4R1:

```
DB2 PREP source_application_program SQLFLAG MVSDB2V41 SYNTAX
```

The SQLFLAG option does not enforce any changes in SQL use; it only issues informational and warning messages regarding syntax incompatibilities in the precompiler listing. If incompatibilities are found, the preprocessing is not terminated abnormally.

Note: For details about the SQLFLAG option, see *DATABASE 2 Command Reference for common servers Version 2*.

3.3 Other DRDA PREP and BIND Parameters

This section summarizes other PREP and BIND parameters that you can use in a DRDA environment.

3.3.1 Binding from a DB2 common server

Table 6 lists some of the DRDA PREP and BIND options that are not fully supported locally by DB2 common server but can be specified when binding to a DB2 for MVS/ESA AS:

.

Table 6 (Page 1 of 2). DRDA PREP and BIND Options of the DB2 common server

Option	Valid Values	Description
VALIDATE	BIND-RUN	Check for object existence and authorizations at bind-time or at run-time. Must use BIND value to DRDA packages and plans.
STRDEL	APOSTROPHE-QUOTE	Designates whether an apostrophe (′) or a quote (″) will be used as a string delimiter within SQL statements.
DECDEL	PERIOD-COMMA	Designates whether a period (.) or a comma (,) will be used as a decimal point indicator in decimal and floating point literals.
ACTION	ADD-REPLACE/RETAIN	Indicates whether the package can be added or replaced.

Table 6 (Page 2 of 2). DRDA PREP and BIND Options of the DB2 common server		
Option	**Valid Values**	**Description**
DEC	31 - 15	Specifies the maximum precision to be used in decimal arithmetic operations.
RELEASE	COMMIT - DEALLOCATE	Indicates whether resources are released at each COMMIT point, or when the application terminates.
DEGREE	1 - deg_of_I/O_par - ANY	Specifies whether or not the query is executed using I/O parallel processing.
DYNAMICRULES	RUN - BIND	Specifies which authorization identifier to use when dynamic SQL in a package is executed. Either the authorization-id of the person executing the package or the package owner. Valid for DB2 for MVS/ESA V4.1.
SQLERROR	NOPACKAGE - CONTINUE	Indicates whether to create a package if SQL errors are encountered.
VERSION	versionid	Specifies the version of the package.
REPLVER	versionid	Specifies the version of the package to be replaced.
EXPLAIN	YES - NO	Is supported and creates entries in the PLAN_TABLE in DB2 for MVS/ESA. If PLAN_TABLE does not exist, or if tables were not yet created, entries are not created, but you do not get any error messages.

3.3.2 Binding from DB2 for MVS/ESA

Table 7 lists the DRDA BIND options available when binding from a DB2 for MVS/ESA AR to a DB2 common server AS.

Table 7 (Page 1 of 2). DRDA BIND Options to DB2 for MVS/ESA		
Option	**Valid Value**	**Description**
VALIDATE	BIND	VALIDATE (RUN) is not supported by DB2 common server AS.

Table 7 (Page 2 of 2). DRDA BIND Options to DB2 for MVS/ESA		
Option	**Valid Value**	**Description**
STRDEL	APOSTROPHE	STRDEL (QUOTE) is not supported by DB2 common server AS.
ACTION	REPLACE/RETAIN(YES)	ACTION (ADD) or (REPLACE with RETAIN NO) is not supported by DB2 common server AS.
DEC	31	DEC (15) is not supported by DB2 common server AS.
RELEASE	COMMIT	RELEASE(DEALLOCATE) is not supported by DB2 common server AS.
DEGREE	1 - deg_of_I/O_par - ANY	Specifies whether or not the query is executed using I/O parallel processing. This option is valid only if connected to a DRDA Level 2 AS. We used DEGREE (1) and DEGREE (ANY), and the bind of the package ended without errors.
DYNAMICRULES	RUN	DYNAMICRULES (BIND) is not supported by DB2 common server AS.
SQLERROR	NOPACKAGE	SQLERROR (CONTINUE) is not supported by DB2 common server AS.
VERSION	null	Any other value is not supported by DB2 common server AS.
REPLVER	null	Any other value is not supported by DB2 common server AS.

Note: EXPLAIN is not supported when binding from DB2 for MVS/ESA to DB2 common server. DB2 common server V2R1 does not support the EXPLAIN YES or NO bind option. To obtain explain output for a package that was bound from a DB2 for MVS/ESA AR to a DB2 common server AS, use the db2expln tool to explain the package after it was bound in the DB2 common server.

The DECDEL (COMMA) precompile option is not supported by DB2 common server AS.

3.4 DB2 for MVS/ESA Sample Programs

DB2 for MVS/ESA is shipped with a series of sample programs. This section describes the following DB2 for MVS/ESA sample programs and explains how to enable them to access a DB2 common server:

- SPUFI

- DSNTIAUL
- DSNTIAD
- DSNTEP2.

3.4.1 SPUFI

SPUFI is a TSO application program that allows interactive SQL access to DB2 common server databases from DB2 for MVS/ESA.

You can use SPUFI to execute an interactive CONNECT statement (interactive does not mean dynamic). The CONNECT statement is valid only if it is embedded within your program as static SQL, although the name of the server can be accepted as a host variable, as SPUFI does.

Figure 54 shows the SPUFI screen. In field 10 you specify the location (SAMPLE). SPUFI then receives SAMPLE as a variable and issues a CONNECT statement to this location.

```
                          SPUFI                              SSID: DB3A
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:        (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> JCL.CNTL(SELORG)
 2  VOLUME SERIAL ... ===>            (Enter if not cataloged)
 3  DATA SET PASSWORD ===>            (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
 4  DATA SET NAME ... ===> OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> NO         (Y/N - Display SPUFI defaults panel
 6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?
 9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===> SAMPLE


PRESS:  ENTER to process    END to exit             HELP for more inform
```

Figure 54. SPUFI Screen

The AS specified at connect time can be a DB2 common server. SPUFI is distributed precompiled with CONNECT(1) (see 5.6, “Precompile Options” on page 103), which implies one single connection within a unit of work.

SPUFI can be bound at several locations (any DRDA AS). A package must be created in each AS.

Figure 55 on page 72 shows a modified installation job (DSNTIJSG) that binds SPUFI packages to location SAMPLE.

```
//DSNTIRU EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(DB3A)
  BIND PACKAGE(SAMPLE.DSNESPCS) MEMBER(DSNESM68) -
       ACTION(REPLACE) ISOLATION(CS) -
       LIBRARY('DSN310.SDSNSAMP') VALIDATE(BIND)
  BIND PACKAGE(SAMPLE.DSNESPRR) MEMBER(DSNESM68) -
       ACTION(REPLACE) ISOLATION(RR) -
       LIBRARY('DSN310.SDSNSAMP') VALIDATE(BIND)
```

Figure 55. Modified Installation Job DSNTIJSG

The bind options and theirs defaults are not always valid in both environments (bind requester and bind server). You must specify VALIDATE (BIND). See 3.3, “Other DRDA PREP and BIND Parameters” on page 68. The collection names DSNESPCS and DSNESPRR are mapped to the creator of the package.

As explained in 3.1.1, “DB2 for MVS/ESA Environment” on page 59 you must include the remote package in the plan available for SPUFI. You can use the BIND command (see Figure 56) as input (SYSTSIN) in the same modified installation job (DSNTIJSG). SAMPLE′s package must be included in these binds. Observe in Figure 56 that the plan includes all packages in the local DB2 for MVS/ESA AS and in all ASs. There is a plan for isolation level CS and another for RR. You can choose which plan to use by using the SPUFI defaults panel.

```
BIND PLAN(DSNESPRR) PKLIST(CENTSJC.DSNESPRR.DSNESM68, -
                           SAMPLEG.DSNESPRR.DSNESM68, -
                           SAMPLE.DSNESPRR.DSNESM68) -
                    ISOLATION(RR) ACTION(REPLACE)
BIND PLAN(DSNESPCS) PKLIST(CENTSJC.DSNESPCS.DSNESM68, -
                           SAMPLEG.DSNESPCS.DSNESM68, -
                           SAMPLE.DSNESPCS.DSNESM68) -
                    ISOLATION(CS) ACTION(REPLACE)
```

Figure 56. BIND PLAN Command

3.4.2 DSNTIAUL

DSNTIAUL is an assembler language sample program that unloads tables into sequential data sets that can later be reloaded into the same DB2 for MVS/ESA table or into another DB2 for MVS/ESA table. It generates as output the DB2 for MVS/ESA commands required by the LOAD utility and the input file for the LOAD utility. The source can be found in the SDSNSAMP library. DSNTIAUL is prepared and run by installation verification program (IVP) job DSNTEJ2A in the SDSNSAMP library.

Note the following:

- You must bind the package for each location where you want to use DSNTIAUL.
- DSNTIAUL does not support the CONNECT statement. You must bind a different plan for each location, using the CURRENTSERVER parameter. This

parameter allows SQL statements in the plan to be executed at the location specified by the CURRENTSERVER parameter value without requiring a CONNECT statement to have been issued (see Figure 57 on page 73).

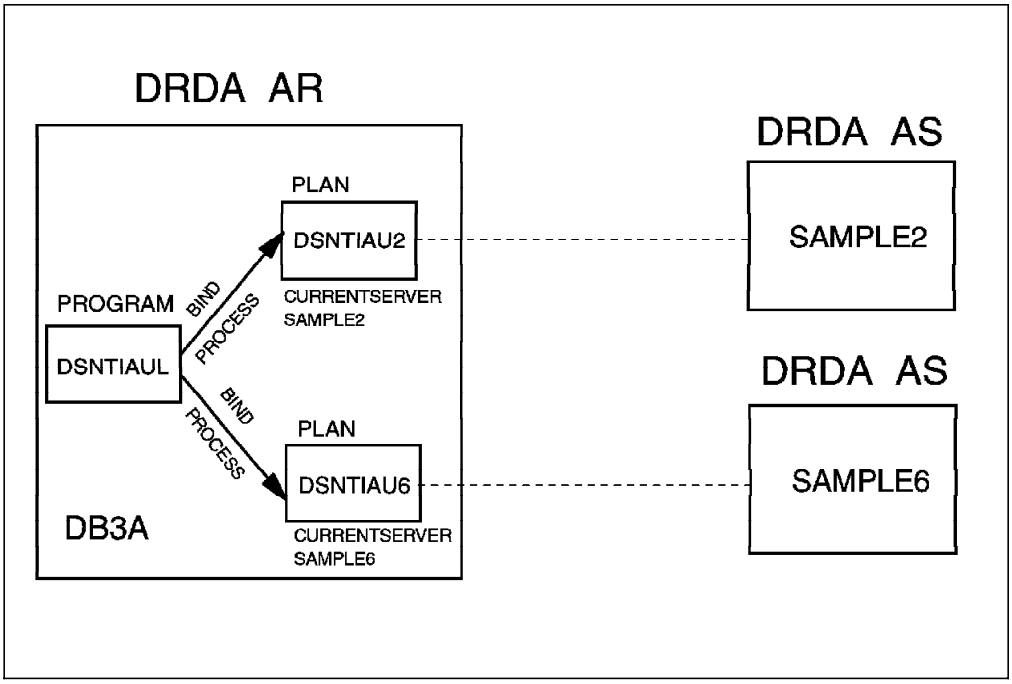


Figure 57. CURRENTSERVER Flow

- For our environment we could not use the PARMS('SQL') in the DSNTIAUL options, where you could specify a SELECT statement in the SYSIN card.

Figure 58 shows sample BIND PACKAGE and BIND PLAN commands for two different locations (SAMPLE2, and SAMPLE6).

```
DSN SYSTEM(DB3A)
BIND PACKAGE (SAMPLE2.DSNTIAUL)            -
             MEMBER(DSNTIAUL) ACT(REP)     -
             ISO(CS) VALIDATE (BIND)       -
             OWNER(SILVIO) QUALIFIER(SILVIO)
BIND PLAN(DSNTIAU2) PKLIST(*.DSNTIAUL.*)   -
                    ACT(REP) ISOLATION(CS)  -
                    SQLRULES(DB2)          -
                    CURRENTSERVER(SAMPLE2)
BIND PACKAGE (SAMPLE6.DSNTIAUL)            -
              MEMBER(DSNTIAUL) ACT(REP)    -
              ISO(CS) VALIDATE (BIND)      -
              OWNER(SILVIO) QUALIFIER(SILVIO)
BIND PLAN(DSNTIAU6) PKLIST(*.DSNTIAUL.*)-
                    ACT(REP) ISOLATION(CS)  -
                    SQLRULES(DB2)          -
                    CURRENTSERVER(SAMPLE6)
END
```

Figure 58. DSNTIAUL BIND Commands

Figure 59 on page 74 shows the sample job to run DSNTIAUL. Note that during execution the program name specified is the same as the name used in DB2 for MVS/ESA, but the plan is specific for the location where the package executes.

```
//PH01S02 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(DB3A)
      RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAU2) -
          LIB('DSN310.RUNLIB.LOAD')
/*
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD SYSOUT=*
//SYSRECOO DD SYSOUT=*
//SYSIN DD *
    ORG WHERE DEPTNUMB >10
/*
//
```

Figure 59. DSNTIAUL Job

Figure 60 shows the output for the example shown for SYSPRINT.

```
DSNT490I SAMPLE DATA UNLOAD PROGRAM
DSNT503I UNLOAD DATA SET SYSRECOO RECORD LENGTH SET TO    51
DSNT504I UNLOAD DATA SET SYSRECOO BLOCK SIZE SET TO    51
DSNT495I SUCCESSFUL UNLOAD            7 ROWS OF TABLE
     ORG WHERE DEPTNUMB >10
```

Figure 60. DSNTIAUL Job

Figure 61 shows the DB2 for MVS/ESA LOAD utility commands placed in SYSPUNCH for the DSNTIAUL job.

```
LOAD DATA LOG NO INDDN SYSRECOO INTO TABLE
    ORG
 (
 DEPTNUMB                              POSITION(       1         )
 SMALLINT                      ,
 DEPTNAME                              POSITION(       3         )
 VARCHAR
      NULLIF(      19)='?',
 MANAGER                               POSITION(      20         )
 SMALLINT
      NULLIF(      22)='?',
 DIVISION                              POSITION(      23         )
 VARCHAR
      NULLIF(      35)='?',
 LOCATION                              POSITION(      36         )
 VARCHAR
      NULLIF(      51)='?'
 )
```

Figure 61. LOAD Command

Figure 62 on page 75 shows the input file (SYSREC00) for the DB2 for MVS/ESA LOAD utility.

```
New England         Eastern      Boston
Mid Atlantic        Eastern      Washington
South Atlantic      Eastern      Atlanta
Great Lakes         Midwest      Chicago
Plains              Midwest      Dallas
Pacific             Western      San Francisco
Mountain            Western      Denver
```

Figure 62. Input File for LOAD Utility

3.4.3 DSNTIAD

DSNTIAD is an assembler language sample program that reads and executes dynamic SQL statements. It can be used to issue all SQL statements except SELECT in batch mode. The installation job DSNTIJTM in the SDSNSAMP library prepares and binds the DSNTIAD plan.

Figure 63 shows an example of the BIND PACKAGE and BIND PLAN commands that enable the use of DSNTIAD at location SAMPLE.

```
DSN SYSTEM(DB3A)
BIND PACKAGE (SAMPLE1.DSNTIAD) -
              MEMBER(DSNTIAD)  -
              ACT(REP) ISO(CS) -
              VALIDATE (BIND)
              OWNER(SILVIO) QUALIFIER(SILVIO)
BIND PLAN(DSNTIAD1)               -
          PKLIST(*.DSNTIAD.*)     -
          ACT(REP) ISOLATION(CS)  -
          SQLRULES(DB2)           -
          CURRENTSERVER(SAMPLE)
END
```

Figure 63. DSNTIAD BIND Commands

Note the following:

- You must bind the package for each location where you want to use DSNTIAD.
- You must bind a different plan for each location, using the CURRENTSERVER option.
- DSNTIAUL does not support the connect statement.

3.4.4 DSNTEP2

DSNTEP2 is a sample program written in PL/I and provided in source form. It executes SQL statements in batch mode. Program DSNTEP2 handles both DDL and DML SQL statements (including SELECT). In addition, support is provided for the static SQL statements of CONNECT, SET CONNECTION, and RELEASE. As DSNTEP2 supports these statements, you need just one plan.

The DSNTEP2 source program and DSNTEJ1P sample job to prepare and bind the DSNTEP2 can be found in the SDSNSAMP library.

Here is an example of how to bind the package and the plan:

```
BIND PACKAGE (SAMPLE.DSNTEP2) MEMBER(DSNTEP2) ACT(REP) ISO(CS) -
             VALIDATE (BIND) OWNER(DB2V2) QUALIFIER(DB2V2)

BIND PLAN(DSNTEP31)  PKLIST(*.DSNTEP2.*) ACT(REP) ISO(CS) SQLRULES(DB2)
```

DSNTEP2 supports DUW (Figure 64).

```
//DSNTEP2  EXEC PGM=IKJEFT01
//JOBLIB   DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLILINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMLINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLIBASE,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMBASE,DISP=SHR
//DBRMLIB  DD  DSN=DSN310.DBRMLIB.DATA,
//             DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(DB3A)
 RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP31) -
      LIB('DSN310.RUNLIB.LOAD')
 END
//*
//SYSIN    DD *
    CONNECT TO SAMPLE;
    SELECT * FROM ORG;
    CONNECT TO SAMPLE6;
    SELECT * FROM ORG;
    SET CONNECTION SAMPLE;
    SELECT * FROM ORG;
    CONNECT RESET;
    SELECT * FROM SYSIBM.SYSLUNAMES;
    RELEASE SAMPLE;
    SET CONNECTION SAMPLE;
    SELECT * FROM ORG;
/*
//
```

Figure 64. DSNTEP2 DUW Support

Note: There are some COMMIT restrictions when you use DSNTEP2.

3.4.5 Binding DB2 common server Utilities

DB2 common server is shipped with the following utilities: import, export, reorg, the CLP, DB2 CLI, and binder program. The utilities have bind files that must be bound to each DRDA AS in which they are used. These bind files are grouped together in different files with extension *.lst*, with each list being specific to a DRDA AS platform.

Table 8 on page 77 shows the DRDA AS platforms and their associated *.lst* file. The list files are in the \SQLLIB\BND directory. See Appendix C, in the *DATABASE 2 Installing and Using OS/2 Clients*, or in the *DATABASE 2 Installing and Using AIX Clients* for a list and description of all of the bind files included in each of the *.lst* files.

Table 8. Utilities List Files	
DRDA AS	**Bind file list name**
MVS/ESA	ddcsmvs.lst
VSE	ddcsvse.lst
VM	ddcsvm.lst
OS/400	ddcs400.lst
common server	db2ubind.lst, db2cli.lst

Here is an example of binding the utilities packages using a *.lst* file for the MVS/ESA DRDA AS:

```
connect to drdamvs user userid using password
bind @ddcsmvs.lst blocking all grant public
```

First access by common server utilities

If you do not bind the DB2 common server utilities packages into DB2 for MVS/ESA the first time that you use them, DB2 common server automatically binds all of the utilities packages by using *ddcsmvs.lst* list. Because of the underlying bind process, the response will be delayed.

Chapter 4. Security Considerations

In this chapter, we discuss the security considerations for connections between DB2 common server workstations and DB2 for MVS/ESA. We also cover various security scenarios that you may encounter. Another source of information with respect to security in the DRDA environment is *DB2 for MVS/ESA DRDA Server: Security Considerations*.

4.1 Workstation AR Security

Security for DDCS for AIX V2.3 has changed very little from the previous releases. However, for DDCS for OS/2, there have been several significant changes.

In this section we focus on the OS/2 workstation, but the discussion is also applicable to the AIX workstation.

4.1.1 Authentication

Authentication is the process that checks a userid against a password to ensure that the userid is valid. You can specify three types of authentication when cataloging a database in the workstation:

- CLIENT

  The userid and password are validated on the client workstation.

- SERVER

  The userid and password are validated on the server workstation.

- DCS

  The userid and password are validated on the host system.

For local clients, CLIENT and SERVER have the same meaning. For remote clients, SERVER and DCS have the same meaning.

Note: You cannot specify authentication for a remote database through the database director. You must specify authentication through the DB2 command line. That authentication overrides the authentication specified at the instance level. This restriction is expected to be removed in a future release of DB2 common server.

4.1.2 Conversation Security

When a DRDA transaction takes place, it uses an APPC conversation. APPC can apply security at a conversation level and is usually specified when you catalog a side information in CM/2 or SNA Server/6000. However, DDCS overrides the conversation security you specify in the communications product with the security that you specify when you catalog the remote node in the DDCS workstation. You can specify two possible security values in the DDCS workstation when you catalog a node:

- PROGRAM

  Both userid and password are sent to the host for validation.

- SAME

Only the userid is sent to the host. An indication that the userid has already been validated is also sent to the host.

4.1.3 Local Client

A local client is a DDCS client that connects directly from the workstation to the host. With the various combinations of AUTHENTICATION and SECURITY parameters, there are four possible security scenarios. These scenarios are depicted in Figure 65 and described in the subsequent paragraphs.

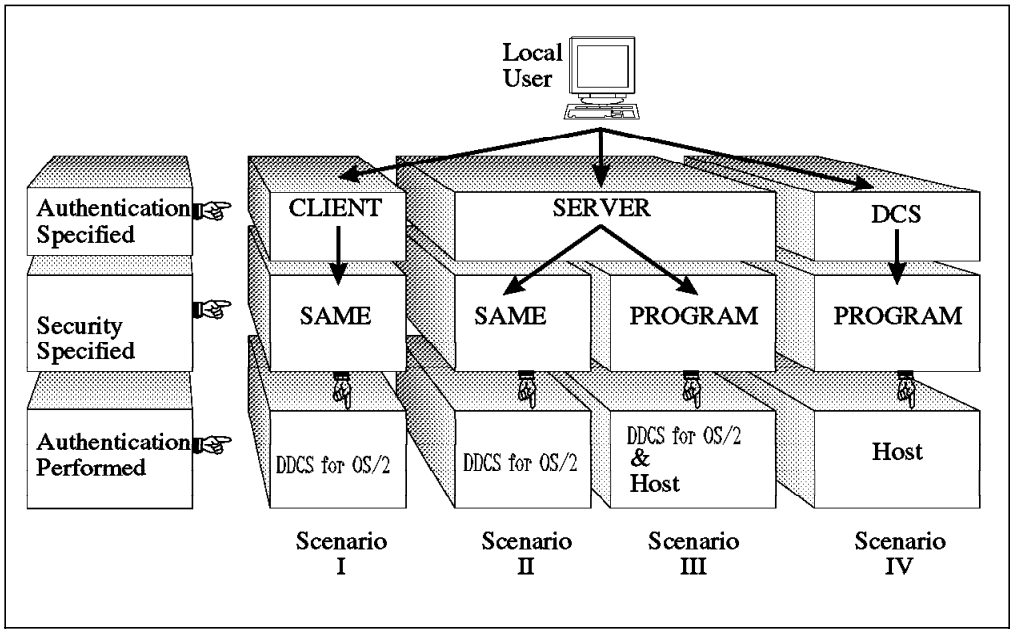


Figure 65. Security for Local Clients

4.1.3.1 Scenarios I and II

Userid and password validation is performed on the workstation only. If the validation succeeds, only the userid is passed to the host, along with an indication that the password already has been validated. If RACF is the security product used on the host, and DB2 for MVS/ESA performs no inbound userid translation, a RACF profile must exist for the userid, but the host does not check the password.

The SECAPT parameter in the APPL macro in VTAM for the destination DB2 system must be set to ALREADYV in these scenarios, and the USERSECURITY column in SYSIBM.SYSLUNAMES must not be equal to C.

On AIX, the userid is only folded to uppercase when transmitted to the host if the login userid belongs to the AIX system group or is root.

APAR PN70160 for DB2 for MVS/ESA folds the userid and password to uppercase before invoking RACROUTE.

Note: Because users can potentially control the userids and passwords on their workstations, there could be a security exposure if these scenarios are used. However, with these scenarios the password is not transmitted over the network, so the risk of password detection is reduced.

4.1.3.2 Scenario III

The userid and password are validated on both the workstation and the host. This implies that the userid and password must be defined on both platforms in the respective security managers.

If the validation succeeds on the workstation, the userid and password are folded to uppercase and passed to the host where they are validated again.

In this scenario, it does not matter how the SECAPT parameter is defined in the APPL macro in VTAM for the destination DB2 system, nor does it matter which value is in the USERSECURITY column in SYSIBM.SYSLUNAMES. Even if SECAPT=ALREADYV or the USERSECURITY column contains an A, the userid and password are still validated on the host.

4.1.3.3 Scenario IV

The userid and password are validated only on the host. Both are folded to uppercase on the workstation and passed to the host for validation. Validation is not performed on the workstation.

In this scenario, it does not matter how the SECAPT parameter is defined in the APPL macro in VTAM for the destination DB2 system, nor does it matter which value is in the USERSECURITY column in SYSIBM.SYSLUNAMES. Even if SECAPT=ALREADYV or the USERSECURITY column contains an A, the userid and password are still validated on the host.

4.1.4 Remote Client

A remote client is a DDCS client that connects to the host through a gateway or server machine. With the various combinations of AUTHENTICATION and SECURITY parameters, on both the client and server, there are four possible security scenarios. These scenarios are depicted in Figure 66 on page 82 and described in the subsequent paragraphs.

Note: We tested these scenarios using the NetBIOS protocol between the client and the gateway. When APPC is used as the protocol between the client and the gateway, some further considerations apply.

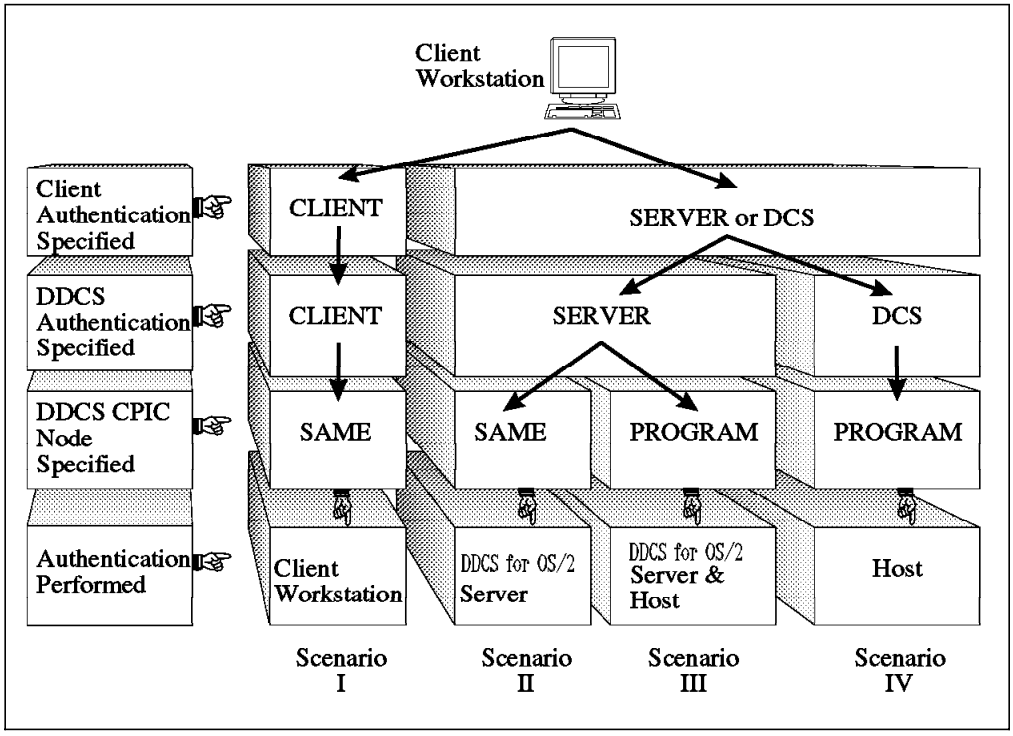


Figure 66. Security for Remote Clients

4.1.4.1 Scenario I

The userid and password are validated on the client workstation which then passes the userid on to the server. In AIX, authentication is not performed at the server, so the userid does not need to be defined in the server workstation. However, in OS/2 the userid must be defined to UPM, but a password does not have to be assigned to that userid. The server passes the userid on to the host where, again, authentication is not performed. If RACF is the security product used by the host, and DB2 for MVS/ESA does not perform inbound userid translation, a RACF profile must exist for the userid, but the host does not check the password.

APAR PN70160 for DB2 for MVS/ESA folds the userid and password to uppercase before invoking RACROUTE.

The SECAPT parameter in the APPL macro in VTAM for the destination DB2 system must be set to ALREADYV in this scenario, and the USERSECURITY column in SYSIBM.SYSLUNAMES must not be equal to C.

Note: Because users can potentially control the userids and passwords on their workstation, there could possibly be a security exposure if this scenario is used. However, with this scenario the password is not transmitted over the network, so the risk of password detection is reduced.

4.1.4.2 Scenario II

The userid and password are passed from the client to the server for validation. The userid and password must be defined to the server workstation, and the password that is passed from the client must match the case of the password stored in the workstation. The userid is folded to uppercase and passed to the host.

If RACF is the security product used by the host, and DB2 for MVS/ESA does not perform inbound user translation, a RACF profile must exist for the userid, but the host does not check the password.

The SECAPT parameter in the APPL macro in VTAM for the destination DB2 system must be set to ALREADYV in this scenario, and the USERSECURITY column in SYSIBM.SYSLUNAMES must not be equal to C.

4.1.4.3 Scenario III

The userid and password are passed from the client to the server for validation. The userid and password must be defined to the workstation, and the password that is passed from the client must match the case of the password stored in the workstation. The userid and password are then folded to uppercase and passed to the host for validation.

If users want to change their password on either the server or on the host, they must update both the workstation and the host password entries.

In this scenario, it does not matter how the SECAPT parameter is defined in the APPL macro in VTAM for the destination DB2 system, nor does it matter which value is in the USERSECURITY column in SYSIBM.SYSLUNAMES. Even if SECAPT=ALREADYV or the USERSECURITY column contains an A, the userid and password are still validated on the host.

4.1.4.4 Scenario IV

This scenario represents how DDCS for OS/2 worked before Release 2.3. The userid and password are passed from the client to the server. The server, however, does not perform any validation. It folds both the userid and password to uppercase and passes them to the host. The host then performs validation on the userid and password.

In this scenario, it does not matter how the SECAPT parameter is defined in the APPL macro in VTAM for the destination DB2 system, nor does it matter which value is in the USERSECURITY column in SYSIBM.SYSLUNAMES. Even if SECAPT=ALREADYV or the USERSECURITY column contains an A, the userid and password are still validated on the host.

4.2 Workstation AS Security

Both DB2 for OS/2 and DB2 for AIX can now perform the role of a DRDA AS. The main issue that you must be concerned with when implementing a DRDA AS in the workstation environment is the authentication level that is set at the instance level. If AUTHENTICATION=CLIENT, a valid userid and password combination is accepted, as well as a userid with an indicator that it has already been validated. If AUTHENTICATION=SERVER, only a valid userid and password combination is accepted.

In CM/2 V1.11 and previous versions, the password in the FMH-5 header cannot be provided to DB2 for OS/2. This forces the OS/2 workstation to use AUTHENTICATION=CLIENT. Connections using CM2 V1.10 and DB2 for OS/2 V2.1 in our testing environment failed when using AUTHENTICATION=SERVER.

4.2.1 Partner LU Considerations for OS/2

If the USERNAMES column in SYSIBM.SYSLUNAMES on the MVS system is O or B, you do not need to catalog a partner LU in CM/2. If you have a partner LU cataloged, DDCS ignores the value that you have specified in the conversation security verification checkbox (see Figure 67).

If the USERNAMES column does not contain O or B, a partner LU must be defined in CM/2, and the conversation security verification checkbox must be checked (see Figure 67).

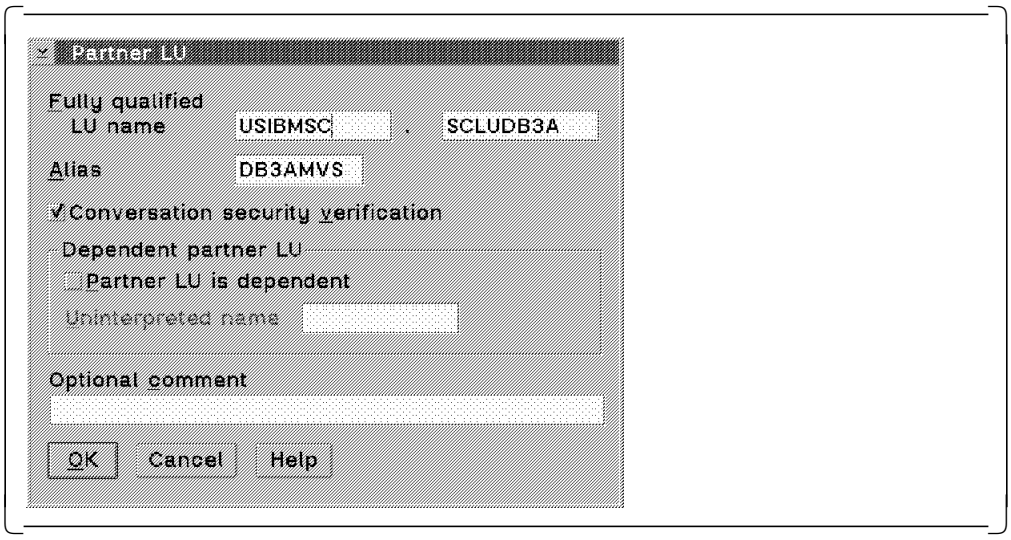


Figure 67. Partner LU Security in OS/2

4.2.2 Partner LU Considerations for AIX

If the USERNAMES column in SYSIBM.SYSLUNAMES on the MVS system is O or B, you do not need to configure a partner LU in SNA Server/6000. If you have a partner LU configured, you should specify for conversation security level already verified or conversation. have specified for conversation security level.

If the USERNAMES column does not contain O or B, a partner LU must be configured in SNA Server/6000, and the conversation security level must be set to already_verified (see Figure 68 on page 85).

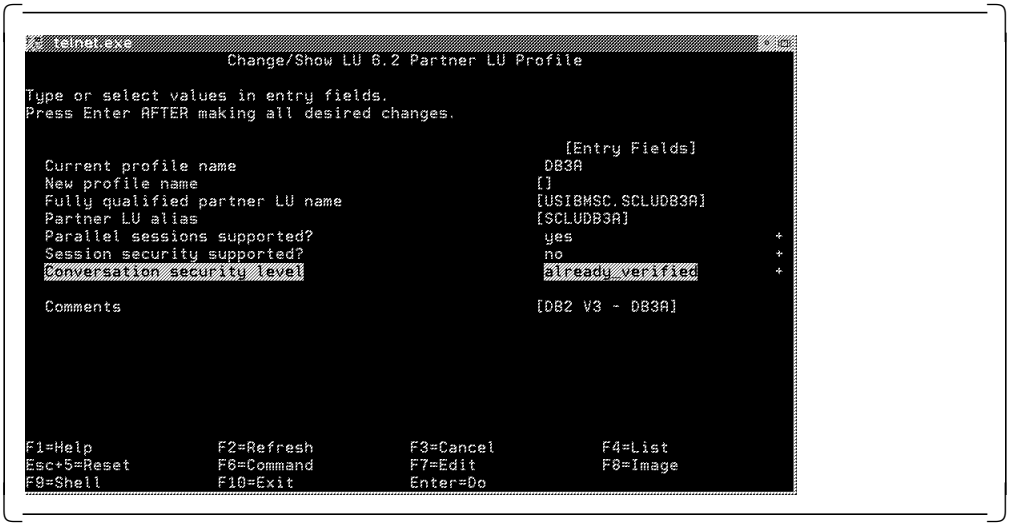


Figure 68. Partner LU Security in AIX

Table 9 is a matrix of the various AS security scenarios.

Table 9. DB2 common server AS Security Scenarios

USERNAMES Column in SYSLUNAMES	Authentication	Already Verified Allowed from Partner LU	Connection Success	
			OS/2	AIX
O or B	Client	No	Yes	Yes
O or B	Client	Yes	Yes	Yes
O or B	Server	No	No	Yes
O or B	Server	Yes	No	Yes
	Client	No	No	No
	Client	Yes	Yes	Yes
	Server	No	No	No
	Server	Yes	No	No

Figure 69 on page 86 depicts the logic flow for the security process in DB2 common server.

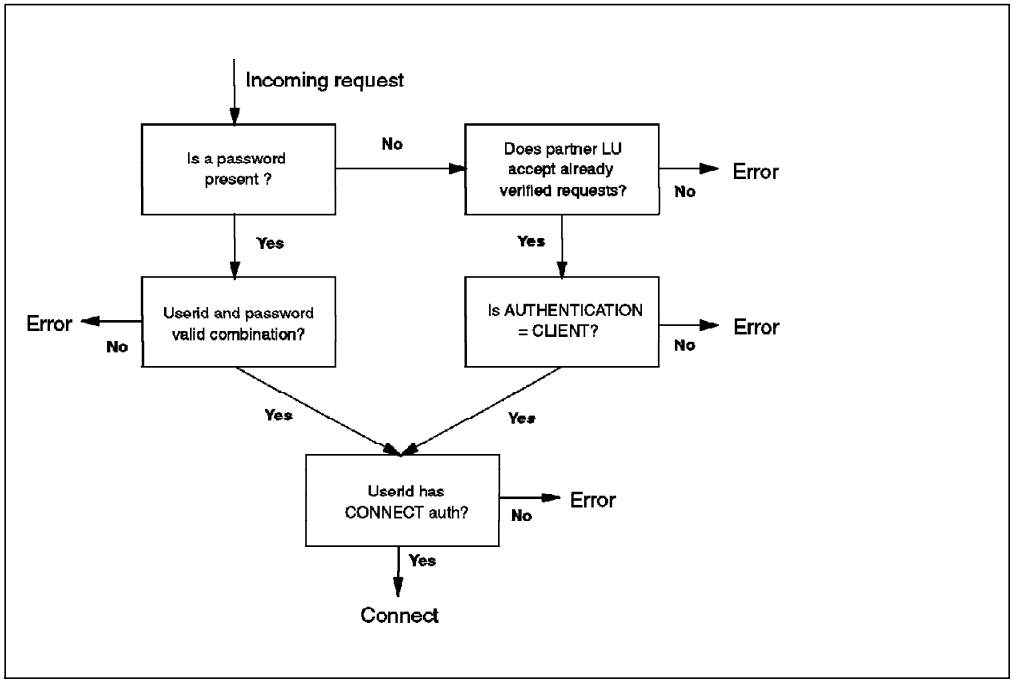


Figure 69. Security Process for DB2 common server

In summary, if you have outbound translation at the host (that is, you are sending both a userid and a password), your partner LU does not have to accept already verified requests. If you do not have outbound translation, you must define the partner LU and specify that it accept already verified requests. You also must specify that AUTHENTICATION=CLIENT for this DB2 instance.

4.3 Host Security

This section provides you with a brief overview of the security environment for DB2 for MVS/ESA. There have been no major changes to the way in which DB2 for MVS/ESA manages security.

DB2 for MVS/ESA does not have the same concept of authentication as the workstation environment. It does, however, allow you to specify a form of conversation security, albeit indirectly. As covered in 2.8, “DB2 for MVS/ESA in DRDA Environment” on page 51, DB2 for MVS/ESA uses the CDB tables to control the communications and security for DRDA transactions (see Figure 48 on page 55).

4.3.1 DB2 for MVS/ESA DRDA AS

Inbound security for a DRDA transaction is controlled using the LUNAME, USERSECURITY, and USERNAMES columns in the SYSIBM.SYSLUNAMES table. In all cases, DB2 for MVS/ESA checks the USERNAMES column. If the value in the USERNAMES column is an I or B, and the incoming LU matches the value in the LUNAME column, DB2 for MVS/ESA substitutes the corresponding userid and password from the SYSIBM.SYSUSERNAMES table in subsequent security checks.

The USERSECURITY field has two valid values:

C Conversation. Any incoming requests must have both a userid and password and are validated by RACF.

A Already verified. VTAM must also have SECAPT=ALREADYV in the APPL macro for DB2 specified for this to be a valid option. An incoming request does not require a password, although if a password is sent, it is checked. This is the default; any other character but C is treated as A.

4.3.2 DB2 for MVS/ESA DRDA AR

Outbound security for DRDA transactions is controlled using the USERNAMES and LUNAME columns in the SYSIBM.SYSLUNAMES table. The USERNAMES field has four valid values:

I Inbound. Incoming userids are translated. However, this is of no relevance when DB2 for MVS/ESA is acting as a DRDA AR.

O Outbound. The MVS userid is translated when going to the LU that is specified in the LUNAME column. The outgoing userid and password are taken from the SYSIBM.SYSUSERNAMES table.

B Both. Both inbound and outbound userids are subject to translation.

blank Neither inbound nor outbound userids are translated.

If the USERNAMES column contains an O or B, DB2 uses the concept of SECURITY=PROGRAM; that is, both a userid and a password are sent to the server. In all other cases, DB2 uses the concept of SECURITY=SAME. That is, just a userid and an indication that it has already been validated are sent to the server.

If there is an O or B in the USERNAMES column, there must also be one or more entries in the SYSIBM.SYSUSERNAMES tables for that same LU. The values in the NEWAUTHID and PASSWORD columns are passed to the remote DRDA server for validation.

4.4 Case Sensitivity Issues

The three platforms that we cover in this redbook (MVS, AIX, and OS/2) have varying degrees of case sensitivity when managing userids and passwords. Thus userid and password case sensitivity is an important issue when dealing with security among the various platforms.

4.4.1 MVS

The primary product that MVS systems use for security validation is RACF. RACF stores all userids and passwords in uppercase and does not perform case translation before the validation check. Therefore, if a userid and password combination is to pass an MVS validity check, they must both be in uppercase before RACF receives them.

APAR PN70160 for DB2 for MVS/ESA folds the userid and password to uppercase before invoking RACROUTE.

4.4.2 OS/2

OS/2 uses UPM as its security manager. UPM stores its userids and passwords in uppercase (even if they were originally entered in lowercase). Upon being given a lowercase userid or password, UPM will fold to uppercase for the purposes of validation.

Note: The exception to this rule is when you are connecting to the host through a gateway. In this case, the password specified in the CONNECT statement must be in uppercase.

4.4.3 AIX

AIX uses a security manager that is built into the operating system itself. The AIX security manager is case sensitive, but, when it receives userids and passwords, it tries several case translation scenarios. Figure 70 shows the logic involved when the AIX security manager receives a userid and password.

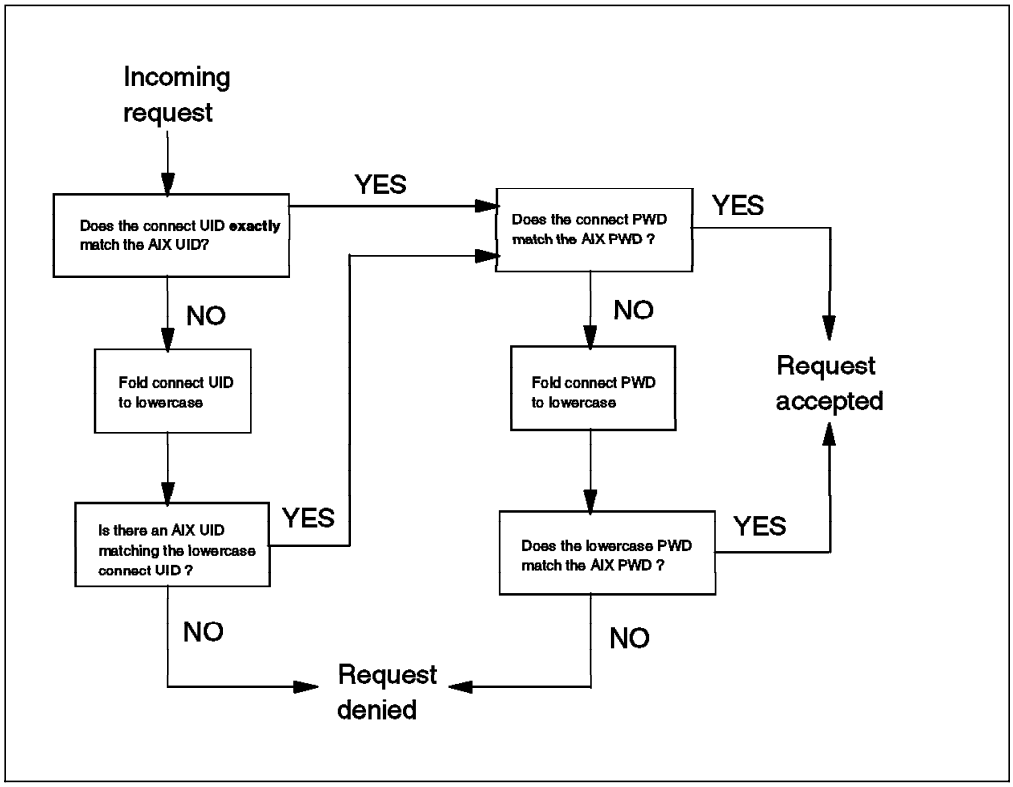


Figure 70. AIX Security Checking

In summary: userids and passwords in the AIX environment are typically in lowercase. The DRDA AS checks for an exact match; if it does not find a match, it folds the userid and password to lowercase and performs another validation check.

4.5 User Profile Management Prompt

DB2 for OS/2 V2.1 has introduced a new environment variable, DB2UPMPR, that controls whether the UPM logon prompt will be displayed if the userid and password combination that is passed to DB2 is invalid. This environment variable is ignored if you specify a userid and password on the command line in the CONNECT statement.

To set the DB2UPMPR environment variable, place one of the following statements in your CONFIG.SYS file:

```
SET DB2UPMPR=ON
SET DB2UPMPR=OFF
```

If DB2UPMPR=ON, the logon prompt is displayed, and the user will have an opportunity to enter a correct userid and password.

If DB2UPMPR=OFF, the logon prompt will not be displayed, and the user will simply receive an error.

If DB2UPMPR is not set, the default action is to display the prompt if an invalid userid and password combination is entered.

Note: In the GA code for DB2 for OS/2 V2.1, this feature is not working as designed. A fix is under construction.

Chapter 5. Distributed Unit of Work

In this chapter we discuss distributed unit of work concepts and parameters. We cover the following topics:

- Remote unit of work concepts
- Distributed unit of work concepts
- DRDA distributed unit of work
- DB2 common server AR
- DB2 for MVS/ESA AR
- SQL extensions
- Precompile parameters
- Distributed unit of work using the CLP.

5.1 Remote Unit of Work Concepts

Remote unit of work (RUW) is included in the first level of DRDA (DRDA1). RUW is a unit of work in which you can connect to only a single AS. Following are the characteristics of RUW:

- One AS per logical unit of work
- Multiple requests in a logical unit of work
- One DBMS in a request
- Application initiation of the commit processing
- Commitment at a single DBMS.

The application issues a CONNECT statement to an AS, such as DB2 for OS/2 or DB2 for MVS/ESA, then it can perform multiple SQL statements within the unit of work. When the application is ready to commit the work, it initiates the commit (one-phase) at the AS that is accessed for this RUW. The application must commit or rollback its current unit of work before it connects to another AS to do additional work. In the next unit of work, the application can access the same AS or a different AS.

5.2 Distributed Unit of Work Concepts

Distributed unit of work (DUW) is part of the second level of DRDA (DRDA2). With DUW, within one unit of work, an application executing in one system can direct SQL requests to multiple remote ASs using the SQL supported by the DBMSs. However, all objects of a single SQL statement are constrained to be in a single AS.

Using DUW, an application can have connections to several ASs in one unit of work. Without DUW, an application could have only one connection to an AS in one commit scope. It is no longer mandatory to end a connection when the unit of work ends. Multiple units of work can be processed during a connection.

DUW allows:

- Multisite read
- Multisite read and single-site update
- Multisite update.

5.2.1 Multisite Read

DUW multisite read has the following characteristics (see Figure 71):

- You can have multiple connections to several ASs.
- Tables can be read from more than one database without closing and reopening cursors.
- The level of commit protocol supported is one-phase commit.
- Updates are not allowed.

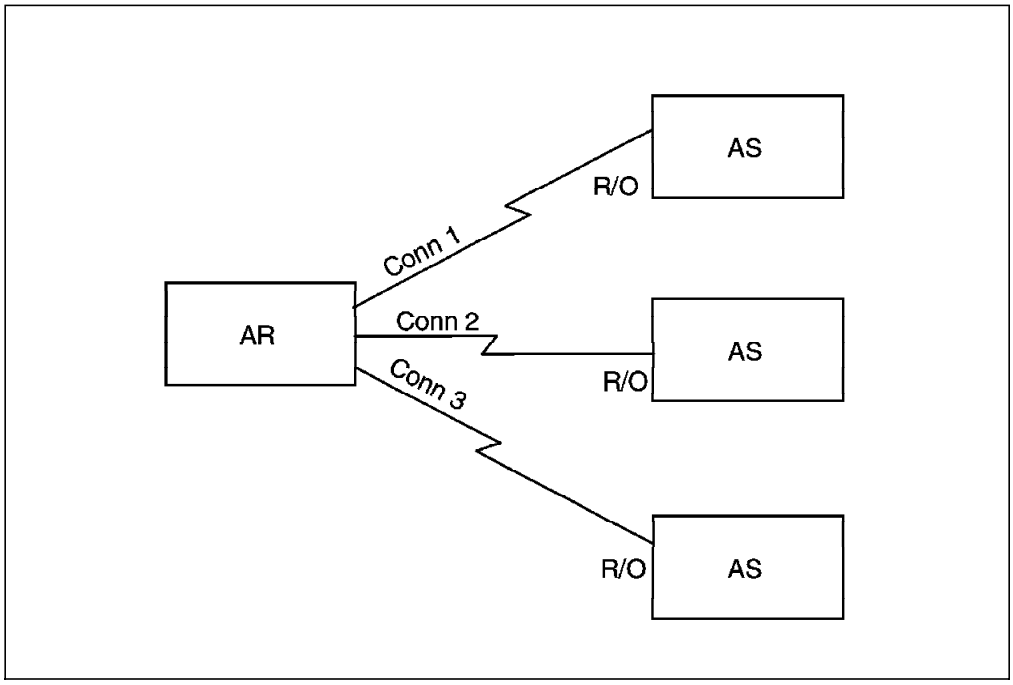


Figure 71. Multisite Read

5.2.2 Multisite Read and Single-site Update

DUW multisite read and single-site update has the following characteristics (see Figure 72 on page 93):

- You can have multiple connections to several ASs.
- Tables can be read from more than one database without closing and reopening cursors.
- The level of commit protocol supported is one-phase commit.
- The services of a syncpoint manager (SPM; see 5.2.4, “DRDA DUW” on page 94) are not required.
- Update is allowed at only one AS.

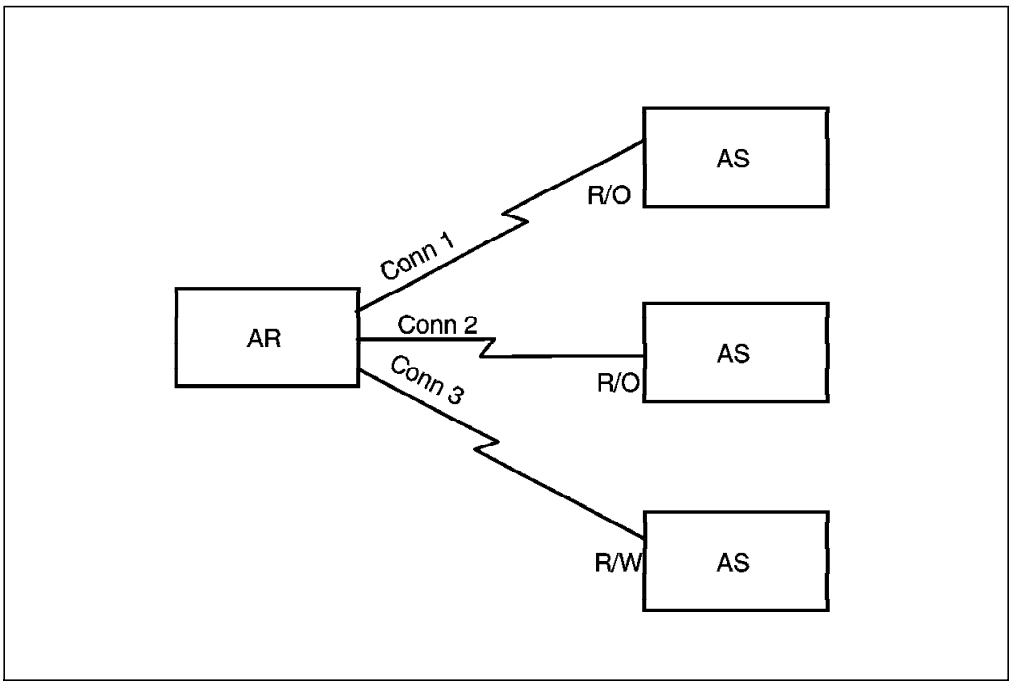


Figure 72. Multisite Read - Single-site Update

5.2.3 Multisite Update

DUW multisite update has the following characteristics (see Figure 73 on page 94):

- You can have multiple connections to several ASs.
- Tables from more than one database can be updated without closing and reopening cursors.
- The level of commit protocol supported is two-phase commit.

  The services of an SPM are required (LU 6.2 two-phase commit protocol).
- Updates to more than one AS are allowed.

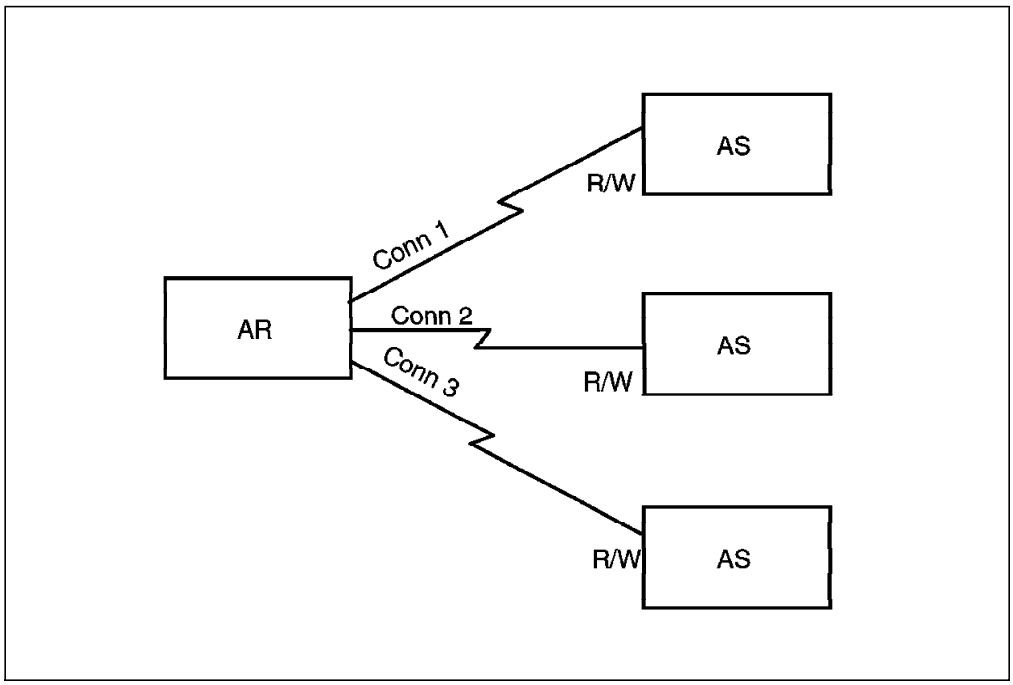


Figure 73. Multisite Update

5.2.4 DRDA DUW

A connection between an AR and an AS is established using an LU 6.2 conversation. Among other characteristics, the SNA LU 6.2 conversation can be protected or unprotected. The conversation is protected or unprotected depending on whether both partners provide the support for an SPM. An SPM is the component of an operating environment that coordinates commit and rollback operations on protected resources. Syncpoint management is used only when the operating environments for both the AR and the AS provide SPM functions. SPMs are required for DRDA multisite update. The enabling of DRDA DUW without multisite update does not require an SPM. The enabling of DRDA DUW with multisite update is based on a protected conversation that supports SPMs.

A relational DBMS can implement DRDA DUW without SPM support, hence without supporting the LU 6.2 two-phase commit protocol. In this case, within a DRDA DUW, the remote DBMS must either be used for read-only or be the only updated site.

Figure 74 on page 95 shows the DRDA commit process executed on a protected conversation; the LU 6.2 two-phase commit protocol is used. The transaction program (TP) on the left-hand side of Figure 74 on page 95 issues a SYNCPT, which the SPM handles in the following manner:

- Phase 1
  - Prepare subphase
    SPM issues prepare to all participants except the last participant. When the request commit has been received from all of them, the next subphase begins.

- Request commit subphase
  The initiator issues a request commit to the last participant and waits for a reply. The reply causes the commit process to enter the phase 2.

• Phase 2

- Committed subphase
  Each participant that did not vote read-only or backout is told of the commit decision and sends a reply.
  SPM completes syncpoint processing by sending committed to all participants except the last, thus entering the forget subphase.
- Forget subphase
  In the forget subphase, SPM waits for forgets from all partcipants execept the last. When all forgets have been received, SPM sends forget to the initiator. Then SPM sends a return code of OK. to the local transaction program (TP).

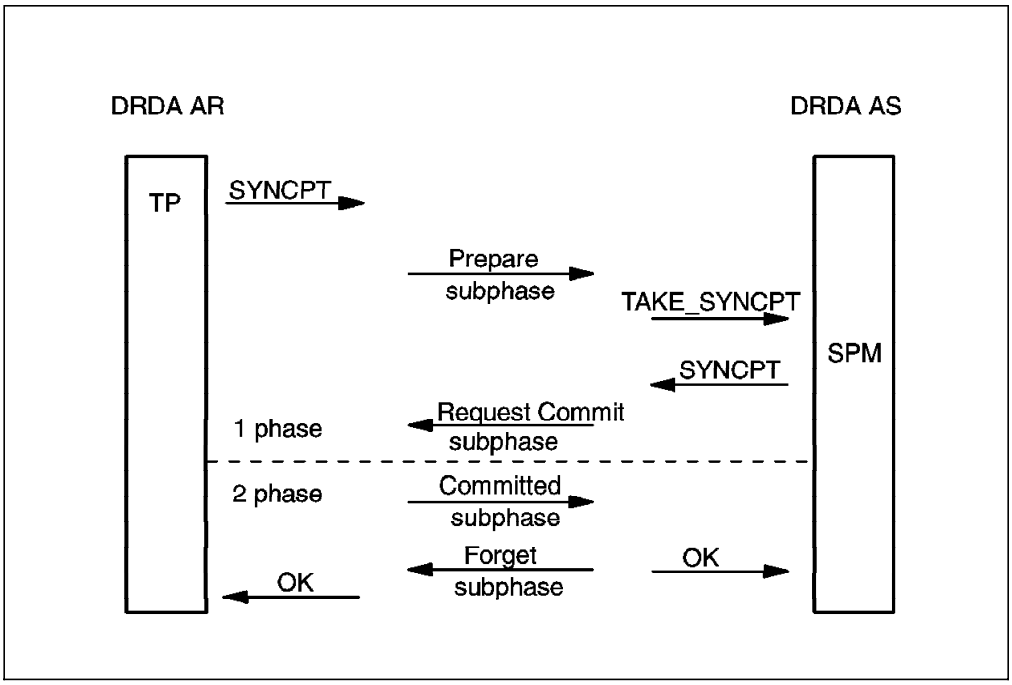


Figure 74. Two-Phase Commit

5.3 DB2 common server AR

DB2 common server provides full support for DUW in a workstation environment, that is, between DB2 for OS/2 and DB2 for AIX.

Although DB2 common server is not DRDA2 compliant (that is, it does not support SPM), it does allow single-site update DUW when connecting to DRDA databases. On the AIX platform DB2 can use the AIX ENCINA products to enable multisite update (see 5.3.2, “With ENCINA Products” on page 100).

5.3.1 Without ENCINA Products

When you use DB2 common server without complementary products such as ENCINA products, DB2 common server uses a transaction monitor to control DUW transactions. In this environment, the server where the first update occurs is important; the server where the first CONNECT occurs is irrelevant.

When a DB2 common server AR connects to a DRDA AS, the AR can update the DRDA AS if, and only if, no other connections exist, or all existing connections are read-only. Otherwise the DB2 common server AR is restricted to read-only for the DRDA AS. A DB2 common server AR maintains its update or read-only status until the unit of work terminates. Once a DB2 common server AS has been been updated, all subsequent connections to any DRDA AS in the unit of work are restricted to read-only. Figure 75 shows the logic flow when connecting from a DB2 common server AR.

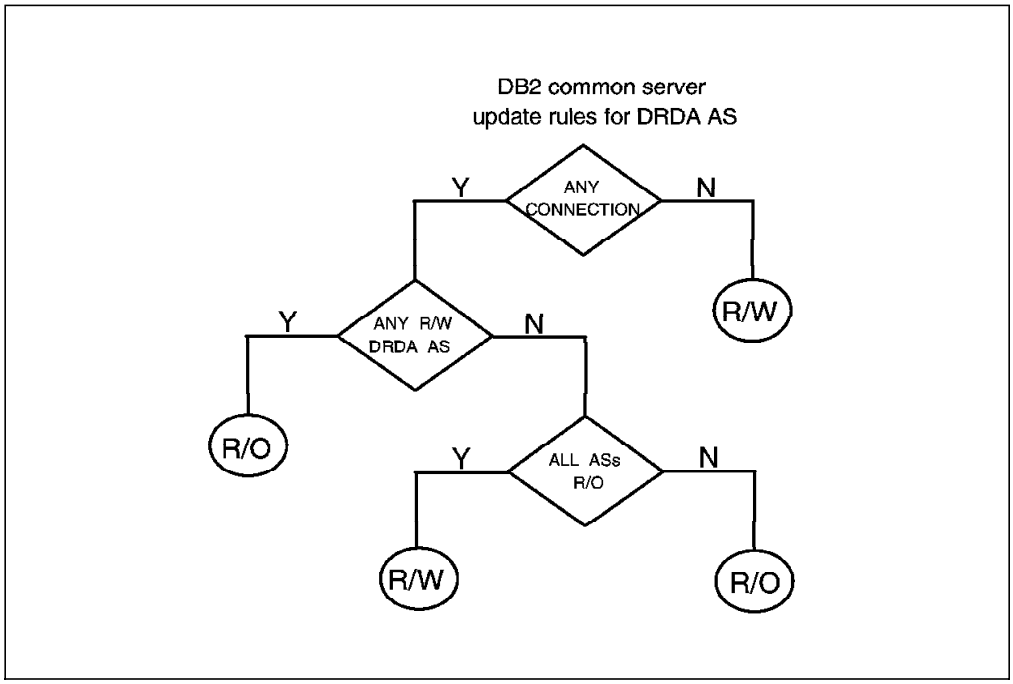


Figure 75. DB2 common server Logic Flow

Figure 76 on page 97 shows single-site update in a DB2 common server AR environment. Requester AR1 (DB2 common server) connects with AS1 (DB2 for OS/2), AS2 (DB2 for MVS/ESA), AS3 (DB2 for AIX), and AS4 (DB2 for OS/400).

If the first update is in AS2 (DRDA2), you can perform updates only in AS2 (R/W). All other ASs (DRDA1 or DRDA2) are in read-only mode.

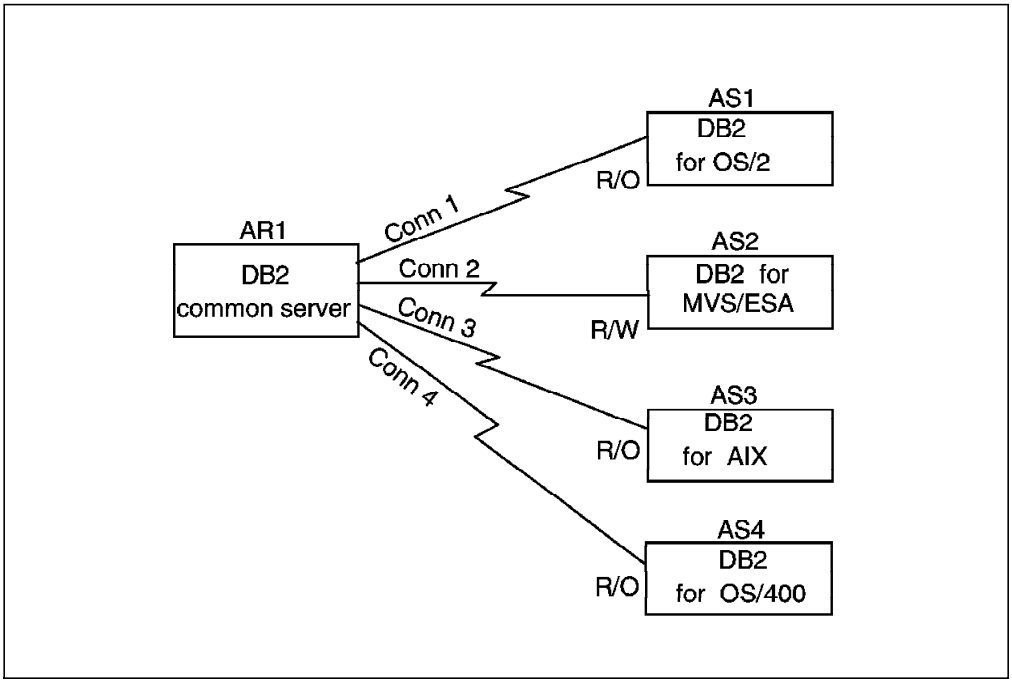


Figure 76. DB2 common server AR in Single-site Update

Figure 77 shows DB2 common server in a multisite update environment. Requester AR1 (DB2 common server) connects with AS1 (DB2 for OS/2), AS2 (DB2 for MVS/ESA), AS3 (DB2 for AIX), and AS4 (DB2 for OS/400).

If the first updater is AS1 (DB2 common server), all DB2 common server ASs (using the private protocol, not DRDA) can be updated (R/W), and all DRDA ASs, such as AS2 and AS4, are in read-only mode.

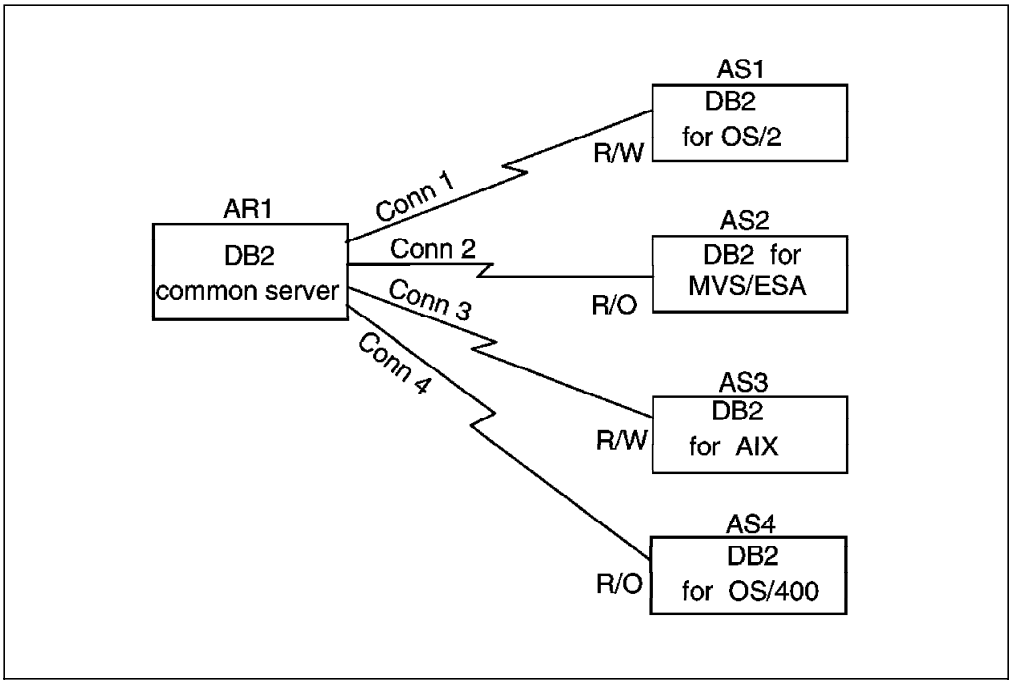


Figure 77. DB2 common server AR in Multisite Update

5.3.1.1 Transaction Monitor

The transaction monitor (TM) in DB2 common server can coordinate multisite updates between DB2 common servers, but the TM does not support LU 6.2 two-phase commit protocol.

DB2 common server provides the TM to coordinate two-phase commit and perform restart and recovery for indoubt transactions. The TM support provided by DB2 common server is equivalent in function to the XA TM (see the *DATABASE 2 AIX/6000 Administration Guide* for more details about the XA TM).

The DB2 common server TM has the following components:

- Coordinator

  The coordinator is the function linked into your application program at compile time. The application program now becomes the coordinator in the two-phase commit process. The application program issues a commit to begin the two-phase commit process.

- Database protection services (DPS)

  DPS is responsible for transaction management within each database. The DPS in each database keeps track of all transactions in progress for that database by using a data structure called a transaction table.

- TM database

  The TM uses the TM database for logging and recovery in the two-phase commit process. The database manager configuration parameter, TM_DATABASE, defines which database alias will be used as a TM database for each DB2 instance. The TM database can be a local or remote database, but it cannot be a DRDA database. It is used as a logger and coordinator. If you specify TM_DATABASE=FIRSTCONNECT, the first non-DRDA database to which your application connects will be used as the TM_DATABASE.

- Resynchronization manager

  The resynchronization manager is responsible for carrying out crash recovery if the two-phase commit process does not complete successfully.

The first phase of the two-phase commit process starts when the application program issues a commit. All resource managers involved in the transaction must be informed that a commit was issued.

The first phase of the two-phase commit process consists of the following steps (see Figure 78 on page 99):

1. The coordinator sends the prepare request to all participating resource managers and wait for the replies.
2. Each resource manager performs a forced write of the prepared log record containing the name of the TM database. This information is used in the event of a crash.
3. Each resource manager informs the coordinator of its status. If the transaction did not perform an update on the resource manager, the resource manager sends a read-only reply. Otherwise, it sends a *yes* if it was successful in the prepare, and a *no* if it was not successful.
4. The coordinator sends a prepare request to the TM database.

5. The TM database performs a forced write of the TM-prepared log record. This information is used in the event of a crash.
6. The TM database informs the coordinator of its status.

If all resource managers respond with a yes vote, the second phase of the two-phase commit begins. Again, referring to Figure 78, the second phase consists of the following steps:

1. The coordinator sends a commit request to all participating resource managers except those that reply with a read-only vote.
2. Each resource manager performs a forced write of the committed log record and commits the transaction.
3. Each resource manager sends an acknowledgment back to the coordinator indicating that the commit was performed successfully.
4. The coordinator sends a commit request to the TM database.
5. The TM database performs an unforced write of the committed log record.
6. The TM database informs the coordinator of its status.

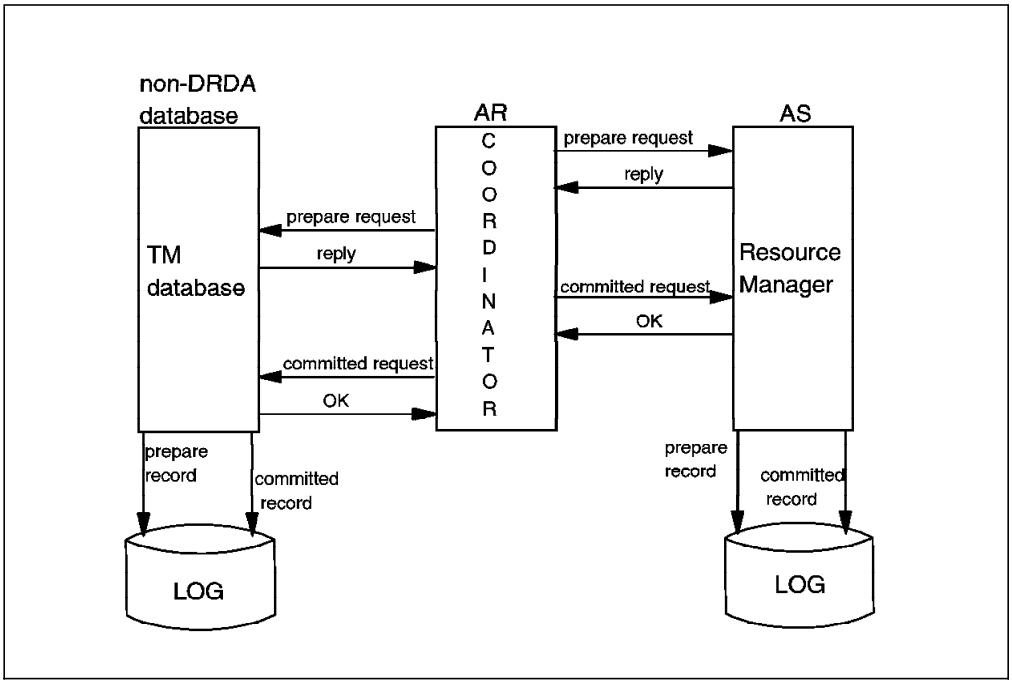


Figure 78. Two-phase Commit Process in Workstations

If anything goes wrong after the first phase, the databases must be resynchronized. Either the TM database or the resource manager intiates the resynchronization. The transactions that finished during the first phase but were unable to commit are called *indoubt transactions*. To list the indoubt transactions issue the following DB2 command:

```
LIST INDOUBT TRANSACTIONS
```

Note: To learn how to resolve indoubt transactions see the *DATABASE 2 Administration Guide for common servers*.

The RESYNC_INTERVAL database manager configuration parameter defines the time interval in seconds for a TM database or a resource manager to retry the recovery of any outstanding indoubt transactions found in the TM database or

the resource manager. This parameter is applicable only to transactions in a multisite update environment.

5.3.2 With ENCINA Products

In a workstation environment the DB2 common server typically can manage only DRDA1 DUW. In AIX, however, the ENCINA products enable multisite update with DRDA2 ASs.

The ENCINA APPC Gateway/SNA provides the support for the LU 6.2 SPM to coordinate the two-phase commit process with DRDA2 ASs. It can work with TP monitor applications such as the ENCINA monitor and CICS/6000.

The ENCINA gateway participates as a communication resource manager (CRM) in a global transaction coordinated by the TP monitor. Conceptually it is like other databases that participate as resource managers. When an application asks to connect to a DRDA2 DBMS, DB2 common server interfaces with the ENCINA gateway to establish a conversation with the DRDA2 DBMS. Then the ENCINA gateway, along with other resource managers, registers with the TP monitor as a participant of a transaction. For DRDA2 ASs, the SPM receives the two-phase commit requests. The ENCINA gateway issues the LU 6.2 syncpoint requests to the SPMs of the DRDA1 ASs (see Figure 79).

To use ENCINA products you must update the database manager configuration parameter, TP_MON_NAME, which identifies the name of the TP monitor product being used. In a CICS environment, set this parameter to CICS; in a non-CICS environment, set this parameter to ENCINA. In all other environments, set this parameter to OTHER.

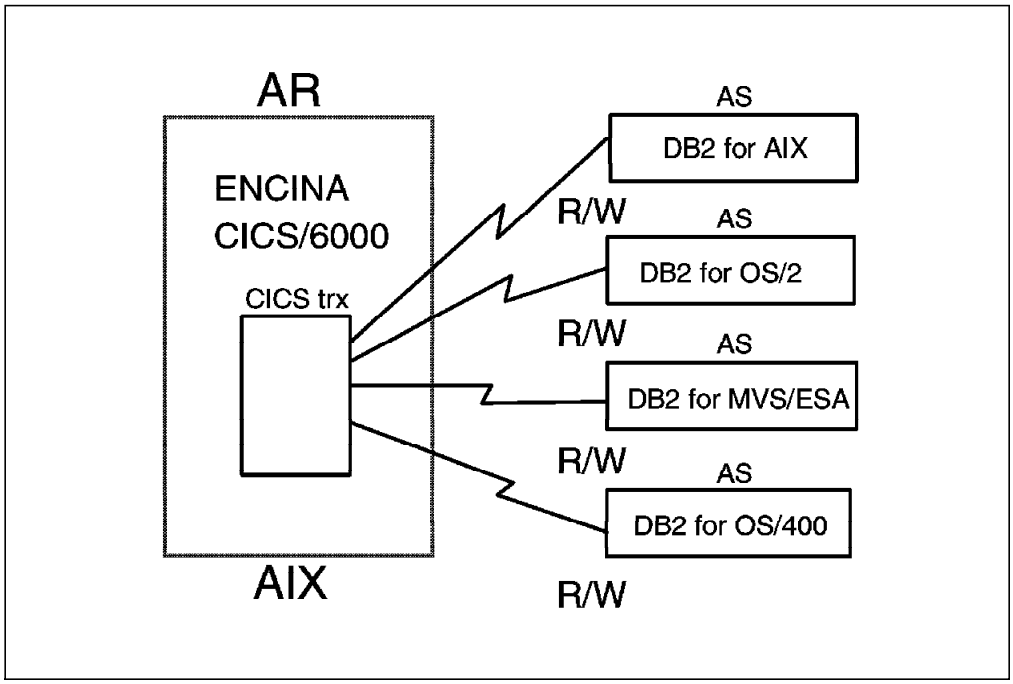


Figure 79. DB2 common server with ENCINA Products

5.4 DB2 for MVS/ESA AR

DB2 for MVS/ESA V3 provides full support for DRDA2.

The first database to which a DB2 for MVS/ESA AR connects determines whether subsequent connections are updatable. If the first database connection is to a DRDA1 AS, all subsequent connections are restricted to read-only. If the first database connection is to a DRDA2 AS, subsequent DRDA2 connections are updatable, and subsequent DRDA1 connections are restricted to read-only.

Note: A DB2 for MVS/ESA AR maintains its update or read-only status until the connection is terminated.

Figure 80 shows requester AR1 (DB2 for MVS/ESA) connected with AS1 (DB2 for OS/2), AS2 (DB2 for OS/400), AS3 (DB2 for AIX), and AS4 (DB2 for MVS/ESA).

If your first connection is to AS1 (DRDA1), you can only perform updates in AS1, and all other ASs (DRDA1 and DRDA2) are in read-only mode.

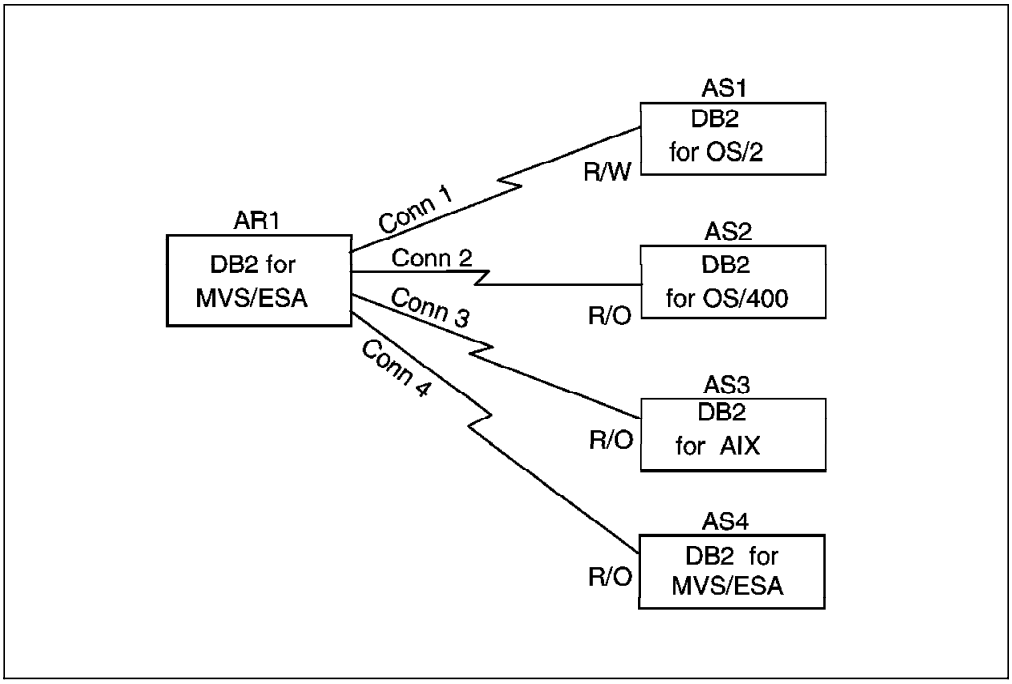


Figure 80. DB2 for MVS/ESA AR in Single-site Update

Figure 81 on page 102 shows requester AR1 (DB2 for MVS/ESA) connected with AS1 (DB2 for MVS/ESA), AS2 (DB2 for OS/400), AS3 (DB2 for OS/2), and AS4 (DB2 for AIX).

If your first connection is to AS2 (DRDA2), all DRDA2 ASs can update (R/W), and all DRDA1 ASs are in read-only mode.

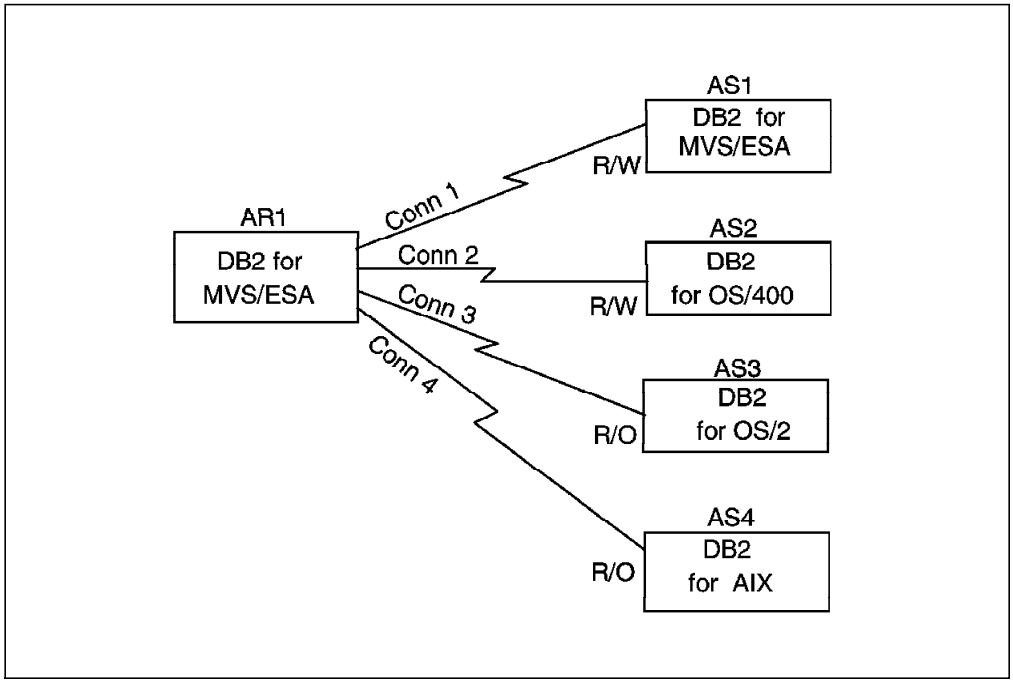


Figure 81. DB2 for MVS/ESA AR in Multisite Update with DRDA2 DBMS

5.5 SQL Extensions

The following SQL statements are enabled when using DUW:

- CONNECT
- RELEASE
- SET CONNECTION
- DISCONNECT
- SET CLIENT
  - CONNECT
  - SYNCPOINT
  - SQLRULES
  - DISCONNECT
- QUERY CLIENT.

These statements can only be embedded within an application program. They are executable statements that cannot be dynamically prepared.

Note: Although you can specify a CONNECT statement that appears to be dynamic in CLP and DSNTEP2, these programs have been written to identify the CONNECT statement and execute using host-variable substitution.

The enhancements related to SQL syntax and semantics include:

- A semantically revised SQL CONNECT statement to establish and/or switch database and location connections. The old behavior is known as a type 1 CONNECT, and the new behavior is known as a type 2 CONNECT (see 5.6, “Precompile Options” on page 103). The application program process

places the previous connection, if any, into a dormant state and connects to the identified AS.

- RELEASE marks a connection as “release pending.” As a result, at commit time the connection is disconnected. RELEASE does not close cursors, does not release any resource, and does not prevent further use of the connection; it merely marks that connection for release.
- SET CONNECTION switches between connected databases. However, it does not establish a connection if a connection does not already exist.
- DISCONNECT explicitly destroys connections. When you use DISCONNECT, you must not have an active unit of work. A DB2 for MVS/ESA AR does not support the DISCONNECT statement.
- SET CLIENT enables an application to change the default connection settings. There must be no existing connections. See 5.6, “Precompile Options” for detailed descriptions of these options:
  - CONNECT specifies whether SQL CONNECT is processed as type 1 or type 2.
  - SYNCPOINT specifies how commits will be coordinated among multiple database connections.
  - SQLRULES specifies whether type 2 CONNECTs should be processed according to DB2 rules or standard rules based on ISO/ANSI SQL92.
  - DISCONNECT specifies which database connections are disconnected at commit.

  A DB2 for MVS/ESA AR does not support the SET CLIENT statement/
- A new QUERY CLIENT statement enables an application to query the current connection settings of the application process. A DB2 for MVS/ESA AR does not support the QUERY CLIENT statement.

5.6 Precompile Options

The semantics of the SQL CONNECT statement and the management of the DUW connections are determined by the following new precompile options of the DB2 common server:

- CONNECT (**1** | 2)
- SYNCPOINT (NONE | **ONEPHASE** | TWOPHASE)
- SQLRULES (**DB2** | STD)
- DISCONNECT (**EXPLICIT** | CONDITIONAL | AUTOMATIC).

5.6.1 CONNECT

Connect type 1 specifies that the application program can use RUW.

Connect type 2 specifies that the application program can use DUW.

This option is supported in the DB2 for MVS/ESA precompile.

At any given time, each connection is in a connected or unconnected state, a held or release-pending state, and a current or dormant state, as we describe below.

Connected and Unconnected States: The connected state is entered when any variation of the CONNECT statement or SET CONNECTION statement is successfully executed to establish or reestablish a current connection.

The unconnected state is entered when an application process does not have a current connection to an AS. In the unconnected state, any variation of the CONNECT statement can be executed, but other SQL statements return an SQLCODE -900. The application enters the unconnected state if an error occurs when a CONNECT statement is issued and no previous connections exist, or if an error occurs in the unit of work, causing the loss of a connection. This state can also be entered when the current connection is disconnected at commit time; that is, it has been marked as “release-pending” earlier in the unit of work.

Held and Release-Pending States: The CONNECT and RELEASE statements control whether a connection is in a held or release-pending state. A held state means that a connection is not to be disconnected at the next commit operation. A connection is placed in the held state by the CONNECT statement.

A release-pending state means that a disconnect is to occur for the connection at the next successful commit operation. A rollback has no effect on connections. Therefore, a release-pending state can be thought of as a pending disconnect. A connection is placed in the release-pending state by the RELEASE statement.

A connection in the release-pending state cannot be placed in the held state, although you can use the SET CONNECTION to reestablish a connection for a release-pending connection. Thus a connection remains in release-pending state across unit-of-work boundaries if a rollback is issued or an unsuccessful commit results in a rollback.

Current and Dormant States: The connection that is in a current state is the one used for SQL statement execution.

A dormant state means that the connection is not current. SQL statements cannot flow to dormant connections, although a COMMIT or a ROLLBACK forces a COMMIT or a ROLLBACK in dormant connections.

The SET CONNECTION and CONNECT statements change the connection for the named server into the current state while any existing connections are either placed or remain in the dormant state. When a dormant connection becomes current in the same unit of work, the state of all locks, cursors, and prepared statements remains the same and reflects their last use when the connection was current.

Table 10 on page 105 compares type 1 and type 2 CONNECTs for the dormant connection states.

Table 10. Dormant Connection States: Type 1 and Type 2 CONNECTs	
Type 1 CONNECT	**Type 2 CONNECT**
Connecting to another AS disconnects the current connection. The new connection becomes the current connection. Only one connection is maintained in an RUW.	Connecting to another AS places the current connection into the dormant state. The new connection becomes the current connection. Multiple connections are maintained in a DUW. If the CONNECT is to an AS in a dormant state, it becomes the current connection.

TYPE 2 restrictions

Connection to a dormant connection using CONNECT (2) is allowed only if SQLRULES(DB2) is specified as a precompile option. If SQLRULES(STD) is specified, the SET CONNECTION statement must be used instead. Otherwise an SQLCODE -842 (connection already exists) is returned.

Note: All other precompile options apply *only* to type 2 CONNECTs. They are ignored for type 1 CONNECTs.

5.6.2 SYNCPOINT

SYNCPOINT specifies how commit and rollback are coordinated among multiple DBMS connections. This option is not supported in DB2 for MVS/ESA.

5.6.2.1 NONE

NONE enables you to update data in multiple DRDA ASs in the same unit of work. The application is responsible for coordination if there is a COMMIT or ROLLBACK failure.

When a COMMIT or ROLLBACK statement is executed, DB2 sends internal COMMITs or ROLLBACKs to all databases that are participating in the unit of work.

The NONE option does not use the two-phase commit process. Error information is logged in the system error log.

If one or more COMMITs fail, an SQLCODE -979 is also returned. The SQLCA indicates the number of failed commits, up to a maximum of five, along with the database alias and SQLSTATE of the failing commits. If a ROLLBACK fails, an SQLCODE -984 is returned.

5.6.2.2 ONEPHASE

ONEPHASE enables you to do multisite read, single-site update to a DRDA or DB2 common server database. The first updated DBMS remains the only updater in that unit of work. Any update attempt to other DBMSs is rejected with an SQLCODE -30090 (SQLSTATE 25000). ONEPHASE allows just one updater, regardless of the DRDA level of the ASs involved.

The ONEPHASE option does not require a TM database or the services of an SPM.

5.6.2.3 TWOPHASE

TWOPHASE is required if you want to use multisite update in DB2 common server databases. TWOPHASE requires the use of a TM process to coordinate the two-phase commit process in the multisite update (see 5.3, “DB2 common server AR” on page 95).

5.6.3 SQLRULES

DB2 rules allow the use of the CONNECT or SET CONNECTION statement to switch from the current connection to a dormant connection.

STD rules allow the use of the CONNECT statement only to establish a new connection, and the application must use the SET CONNECTION statement to switch from the current to a dormant connection.

Note: In DB2 for MVS/ESA you can specify the SQLRULES option in the BIND PLAN command.

5.6.4 DISCONNECT

DISCONNECT specifies which database connections are disconnected at commit. The connections can be those that were explicitly marked with RELEASE (EXPLICIT), those without an open WITH HOLD cursor (CONDITIONAL), or all connections (AUTOMATIC).

Note: In DB2 for MVS/ESA you can specify the DISCONNECT option in the BIND PLAN command.

5.7 Using DUW from the CLP

DB2 common server has two new commands—UPDATE COMMAND OPTIONS and SET CLIENT—that enable you to set various DUW settings when using CLP.

5.7.1 Updating Command Options

To use DUW from the CLP you must turn the auto-commit setting to off. It is also useful to display the SQLCA. Type the following command:

```
DB2 UPDATE COMMAND OPTIONS USING A ON C OFF
```

5.7.2 Updating DUW Options

If you want to do multisite update, you must also specify that you are using CONNECT type 2 for DUW and SYNCPOINT TWOPHASE. Type the following:

- DB2 SET CLIENT CONNECT 2
- DB2 SET CLIENT SYNCPOINT TWOPHASE

Here is the full syntax of the SET CLIENT statement:

```
SET CLIENT {CONNECT {1 | 2}}
           {DISCONNECT {EXPLICIT | CONDITIONAL | AUTOMATIC}}
           {MAX_NETBIOS_CONNECTIONS { 1 | 2 ... 255 }}
           {SQLRULES {DB2 | STD}}
           {SYNCPOINT {ONEPHASE | TWOPHASE | NONE}}
```

By using the UPDATE COMMAND OPTIONS and SET CLIENT commands, you can connect and run commands in a multisite update environment from the CLP. Of course, you are still governed by the same restrictions that apply to application programs in a DUW environment (see 5.3, “DB2 common server AR” on page 95).

The QUERY CLIENT statement displays the options that the CONNECT statement uses. The result of a QUERY CLIENT statement is:

```
The current connection settings of the application
processes are:
                   CONNECT    = 2
                DISCONNECT    = EXPLICIT
  MAX_NETBIOS_CONNECTIONS     = 1
                  SQLRULES    = DB2
                 SYNCPOINT    = TWOPHASE
```

5.7.3 CONNECT Statement

The CLP command, UPDATE COMMAND OPTIONS USING A ON, will make DB2 common server display the SQLCA structure after each SQL statement. When you issue a CONNECT statement and have set that command option, you can examine the SQLCA structure to determine several environmental settings.

Some of the SQLCA fields are:

- SQLERRMC, which is used when a successful connection completes. The field contains such information as code page, userid, database connected, and token database identifier. Here is the token identifier list:

Token	Server
QAS	DB2 for OS/400
QDB2	DB2 for MVS/ESA
QDB2/2	DB2 for OS/2
QDB2/6000	DB2 for AIX
QDB2/HP-UX	DB2 for HP-UX
QOS/2 DBM	ES 1.0 DBM
QSQLDS/VM	SQL/DS VM
QSQLDS/VSE	SQL/DS VSE

- SQLERRD(3), which describes whether the connected database is updatable in the unit of work. The values that can be returned are:
  1. Updatable
  2. Read-only
- SQLERRD(4), which describes the characteristics of the connection. The values that can be returned are:
  1. One-phase commit
  2. One-phase commit; read-only (only applicable to connections to DRDA1 databases in TP monitor environments)
  3. Two-phase commit
  4. Reserved.

In the sample SQLCA structure shown in Figure 82 on page 108, the code page is 850, the userid is USERID, the connected database is SAMPLE, and the database token is QDB2/2.

```
db2 => CONNECT TO SAMPLE USER USERID USING PASSWORD
   Database Connection Information

 Database product       = DB2/2 2.1.0
 SQL authorization ID   = USERID
 Local database alias   = SAMPLE


SQLCA Information

 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: 0   sqlerrml: 38
 sqlerrmc: 1850USERID  SAMPLE  QDB2/24110
 sqlerrp : SQLO2010
 sqlerrd : (1) 0                (2) 0                (3) 1
           (4) 1                (5) 0                (6) 0
 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)
           (7)      (8)      (9)      (10)       (11)
 sqlstate: 00000
```

Figure 82. SQLCA Structure

Note: The SQLCA values described above are not valid in DB2 for MVS/ESA. In DB2 common server you can issue the SET CLIENT and QUERY CLIENT within an application provided that you do not have a current database connection.

Chapter 6. Stored Procedures

In this chapter we present an overview of stored procedures, explain how you can take advantage of this feature, and describe how it is implemented. We also briefly describe the differences between stored procedures and DARI. We cover the way in which stored procedures have been implemented in DB2 for MVS/ESA and examine some sample stored procedures that are provided with the DB2 for MVS/ESA product.

6.1 What Is a Stored Procedure?

A stored procedure is an application program that is stored on the DB2 server and can be invoked by the client using an SQL CALL statement.

In the first two levels of DRDA, all application processing was performed on the client system. The server was responsible only for SQL processing on behalf of the client. A stored procedure enables you to share the application processing between the client and the server.

You can gain many advantages from using stored procedures:

- Network costs are lower because the number of I/O interactions between the client and the DB2 server are reduced.
- The client need not be aware of the server system′s database design.
- The database design is transparent to clients.
- An extra level of protection is provided for sensitive portions of an application program.
- The application is DRDA compliant.
- Maintenance can be performed easily from the server.
- You can take advantage of existing skill sets on the server.

In a typical DRDA application, a network *send/receive* operation is required for most SQL statements. For example, each SELECT (with no blocking), UPDATE, INSERT, and DELETE requires network operations. A single network I/O operation can be performed quickly. If your application issues many SQL calls, however, network traffic can become a major problem. The problem is magnified if you are using a relatively slow network. There is also some overhead associated with building the DRDA request and reply messages. The network I/O operations and the message building overhead combine for an increased SQL operation path length for distributed applications when compared to a local SQL operation (see Figure 83 on page 110).

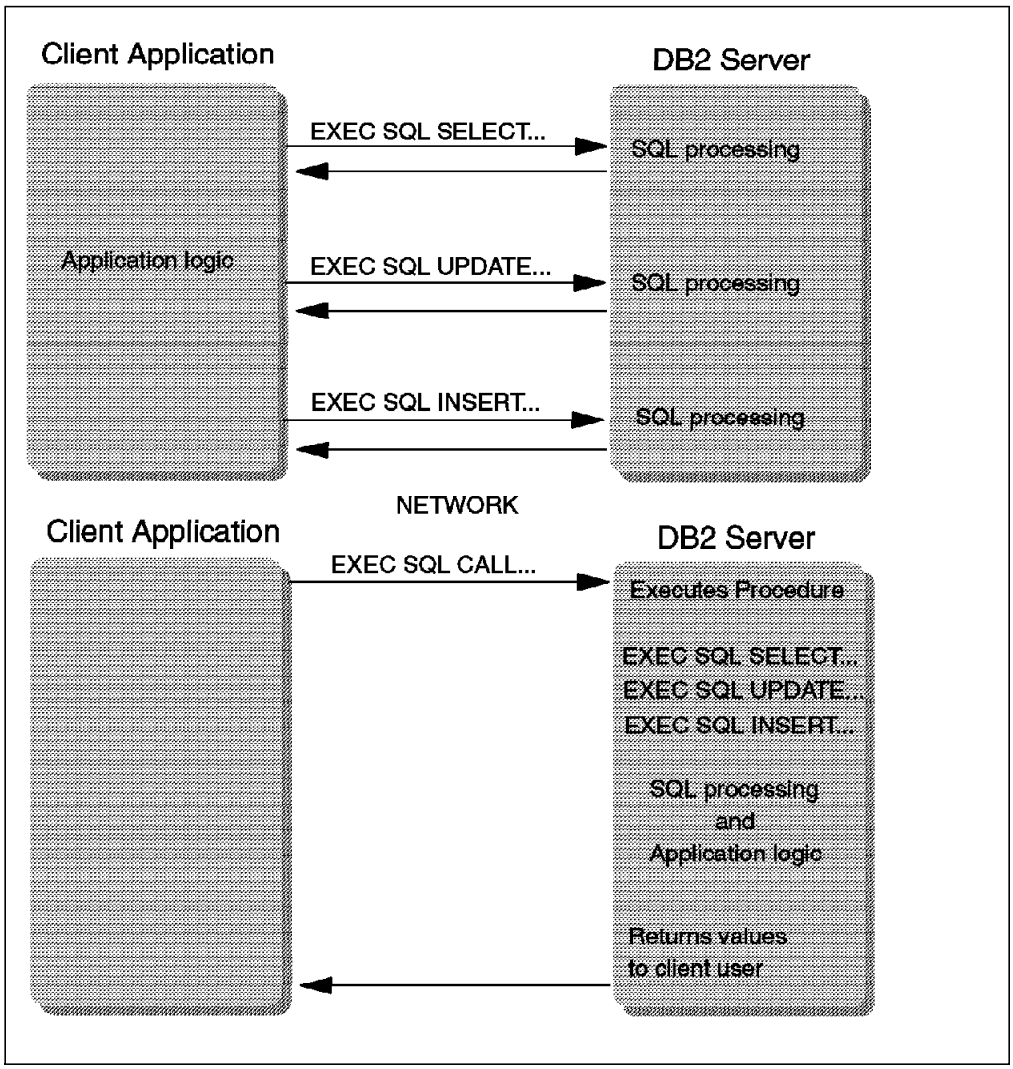


Figure 83. Client/Server Environment with and without Stored Procedure

6.2 Benefits and Limitations

Although stored procedures enable application programmers to use almost any SQL statement to manipulate data, they do have a few limitations:

- Benefits
  - Most DDL, DCL, and DML statements are supported.
  - SQL statements containing three-part names (in DB2 for MVS/ESA) are supported.
  - Execution of DB2 commands is supported.
  - Instrumentation facility interface calls (in DB2 for MVS/ESA) are now supported. This enables inspection of the DB2 trace records.
  - Stored procedures also enable access to non-DB2 resources, such as data sets.

- Limitations
  - Cannot call another stored procedure. The SQL CALL statement is invalid within a stored procedure.
  - Responsibility for the COMMIT belongs to the client. The SQL COMMIT statement is invalid within a stored procedure.
  - SQL ROLLBACK is allowed, although it does not take place immediately. The unit of work is placed in a “must rollback” state, which forces the client to issue a ROLLBACK
  - In DB2 for MVS/ESA only stored procedures written in PL/I, C, Assembler, or COBOL can be invoked because they run in an LE/370 environment. Stored procedures can, however, invoke other programs written in different languages. The program on the AR that invokes the stored procedure can be written in any language that the precompiler supports.
  - In DB2 common server, stored procedures written in REXX are also supported.
  - There is no SYSIBM.SYSPROCEDURES catalog table in DB2 common server. This forces the client to know about the physical location of the stored procedure that is to be invoked.

    Note: You can define a table called DB2CLI.PROCEDURES in which you can keep details about stored procedures and reference later using the call level interface (CLI). However, CLI uses this table as a reference only. The table does not have to exist, nor does it have to be populated in order to invoke stored procedures.
  - Stored procedures require DB2 for MVS/ESA V4 or later.

6.3 SQL CALL Authorization and Syntax

The SQL CALL statement is an executable statement and as such can only be embedded in an application program. It cannot be dynamically prepared. It can, however, use a host variable to store the stored procedure name.

In this section we cover the authorizations necessary to invoke a stored procedure and the syntax of the SQL CALL command.

6.3.1 Authorization

The authorization rules to invoke a stored procedure vary according to the server at which the stored procedure is stored.

6.3.1.1 DB2 common server

The privileges required by the authorization ID of the CALL statement at **run-time** must include at least one of the following:

- EXECUTE on the stored procedure package
- CONTROL on the stored procedure package
- SYSADM or DBADM authority.

6.3.1.2 DB2 for MVS/ESA Server

The privileges required by the authorization ID of the CALL statement at **bind-time** must include at least one of the following:

- EXECUTE on the stored procedure package
- Ownership of the stored procedure package
- PACKADM for the package's collection
- SYSADM authority.

6.3.2 Syntax

Figure 84 and Figure 85 show the syntax of the SQL CALL statement for the DB2 common server and DB2 for MVS/ESA, respectively.

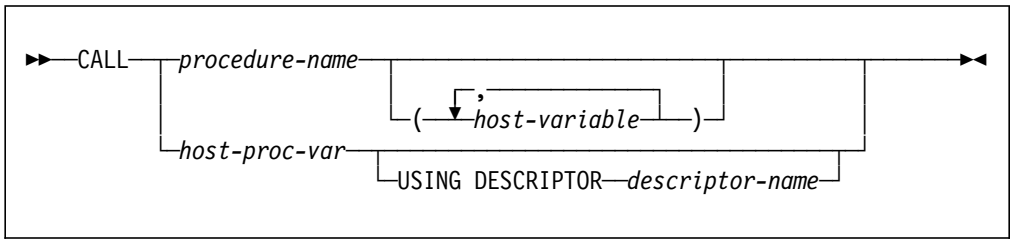


Figure 84. SQL CALL Statement Syntax: DB2 common server

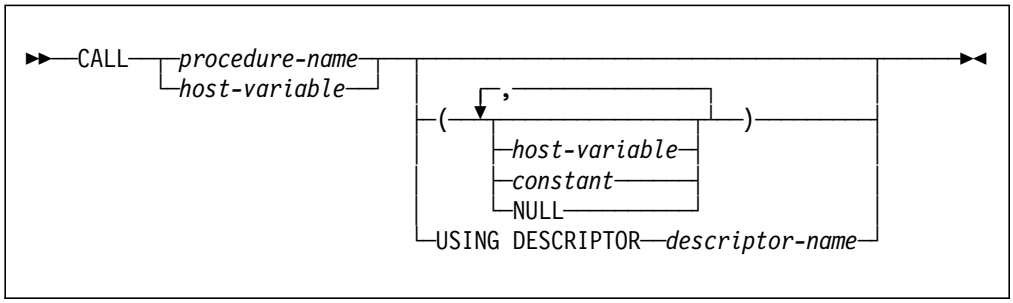


Figure 85. SQL CALL Statement Syntax: DB2 for MVS/ESA

Refer to *DATABASE 2 SQL Reference for common servers* and *DB2 V4 Application Programming and SQL Guide* for more information about the syntax and usage of the SQL CALL statement.

6.4 DARI and SQL CALL

Because the way in which parameters are passed differ for DARI and SQL CALL, it is unlikely that any of the existing DARI procedures can be invoked by using the new SQL CALL statement. The old sqleproc() function still must be used for existing DARI procedures.

However, a newly developed DARI client/server application can be written such that it can be invoked by both the DARI API and the SQL CALL statement. As long as the DARI API does not use the special input parameter to pass additional information to the server procedure and uses only one SQLDA for input and output, at run time DARI and the SQL CALL statement will have the same flow. Table 11 on page 113 summarizes the major differences between the DARI API and the SQL CALL statement.

Table 11. DARI API and SQL CALL	
DARI API	**SQL CALL**
Can invoke DARI procedures	Can invoke DARI procedures
Cannot invoke stored procedures	Can invoke stored procedures
Cannot use DRDA	DRDA compliant
SQLCA explicit	SQLCA implicit
No SQL statements sqleproc() is a run-time call	SQL CALL statement Stored statically at bind time, invoked at run time
One SQLDA for input, one SQLDA for output	One SQLDA for both input and output
Procedure name is a literal in the parameter list	Procedure name can be specified as a literal or as a run-time value in a host variable

The DARI API cannot be used in conjunction with DDCS to invoke a DRDA server's stored procedures. The SQL CALL statement, however, can invoke stored procedures locally using the DB2 private protocol, or remotely using the DRDA protocol. The stored procedures can be invoked on servers such as DB2 common server, DB2 for MVS/ESA, and DB2 for OS/400.

The SQLCA is an explicit parameter in the DARI API. It is returned implicitly by the SQL CALL statement.

No SQL statements are used when invoking DARI. The sqleproc() is a run-time call. CALL is the SQL statement that is used to invoke the stored procedure. It must be contained in the SQL application's bind file.

The DARI special input parameter is a character string for additional information that you specify to the sqleproc() function, along with other SQLDA information. There is no equivalent in the SQL CALL statement. Additional information cannot be passed to the stored procedure.

When using DARI, the procedure name must be literally specified in the sqleproc() parameter list. In the SQL CALL statement, however, the procedure name can be passed by using host variable substitution.

6.5 Stored Procedures in DB2 for MVS/ESA

An application program uses the SQL CALL statement to invoke a stored procedure (see Figure 84 on page 112 and Figure 85 on page 112 for the syntax of SQL CALL). When a stored procedure is invoked in DB2 for MVS/ESA, the following steps are executed:

1. DB2 for MVS/ESA is notified that the client wants to run a stored procedure and determines the stored procedure name.
2. DB2 for MVS/ESA searches the SYSIBM.SYSPROCEDURES catalog table for a row associated with the stored procedure name. SYSIBM.SYSPROCEDURES identifies the load module to be used for the stored procedure, as well as its parameter list requirements. The information retrieved from SYSIBM.SYSPROCEDURES is cached, so that

subsequent SQL CALL statements for a previously called stored procedure do not have to perform I/O to the catalog again.

3. DB2 for MVS/ESA finds a task control block (TCB) in the DB2 stored procedure address space (xxxSPAS) that is available for work. DB2 for MVS/ESA requests xxxSPAS to run the stored procedure. xxxSPAS uses the DB2 thread of the user that issued the SQL CALL statement to invoke the stored procedure.
4. Using the information stored in SYSIBM.SYSPROCEDURES, xxxSPAS loads and runs the load module associated with the stored procedure. The load module may stay resident in the xxxSPAS region for subsequent invocations, depending on the value in the STAYRESIDENT column.
5. The stored procedure module gains control and has access to a parameter list containing the input and output parameters provided on the SQL CALL statement. The stored procedure can use these parameters to issue SQL statements and/or perform any other MVS operation that can be performed from an application program, such as invoking batch programs or manipulating VSAM data. Any DB2 locks or updates created by SQL statements within the stored procedure become part of the caller's unit of work.
6. After the stored procedure terminates, DB2 for MVS/ESA copies the output parameters into the calling application's parameter area and returns to the calling program.
7. The calling program takes action on the values returned and eventually performs a COMMIT. At that time, all SQL operations performed by the stored procedure are committed.

6.5.1 DB2 for MVS/ESA Catalog Table

When a stored procedure is invoked, DB2 for MVS/ESA checks the SYSIBM.SYSPROCEDURES catalog table (see Table 12), before it passes the thread to a xxxSPAS TCB.

Table 12 (Page 1 of 2). SYSIBM.SYSPROCEDURES Table

Column Name	Description
PROCEDURE	Contains the name of the stored procedure specified in the SQL CALL statement.
AUTHID	Links the SQL authorization ID of the user invoking the SQL CALL to the stored procedure name.
LUNAME	Links the LU name of the user invoking the SQL CALL to the stored procedure name.
LOADMOD	The member name of the MVS load module that DB2 should load to satisfy the request for the stored procedure.
LINKAGE	The linkage convention used to pass parameters to the stored procedure (determines whether or not NULL input parameters are allowed).
COLLID	The name of the collection to be used when the stored procedure is invoked.

Table 12 (Page 2 of 2). SYSIBM.SYSPROCEDURES Table	
Column Name	**Description**
LANGUAGE	The programming language used to create the stored procedure. Possible values are ASSEMBLE, PLI, COBOL, and C.
ASUTIME	The number of CPU service units permitted for any single invocation of this stored procedure. If this number is exceeded by the execution of the stored procedure, the stored procedure is canceled by DB2.
STAYRESIDENT	Determines whether the stored procedure load module is deleted from memory when the stored procedure ends.
IBMREQD	Determines whether the row came from the basic machine readable material (MRM) tape.
RUNOPTS	The LE/370 run-time options to be used for this stored procedure. LE/370 is a prerequisite for stored procedures.
PARMLIST	Defines the parameter list expected by the stored procedure.

6.5.2 DB2 for MVS/ESA Sample

DB2 for MVS/ESA V4 provides a sample stored procedure called DSN8EP2. It is a PL/I program that uses the IFI to process a DB2 command that is passed in by a program issuing an SQL CALL statement. The relevant PL/I statements in DSN8EP2 are:

```
DSN8EP2: PROCEDURE(INPUTCMD, IFCA_RET_HEX, IFCA_RES_HEX, IFCA_BYTES_MOVED, RETURN_BUFF);

CALL DSNWLI (FUNCTION, IFCA, RETURN_AREA, OUTPUT_AREA);
```

DB2 for MVS/ESA V4 also provides a C program, DSN8ED1, which issues an SQL CALL statement to invoke DSN8EP2. The relevant C statements in DSN8ED1 are:

```
EXEC SQL CONNECT TO :db2loc2;
EXEC SQL CALL DSN8EP2(:stmtbuf,
                      :ifca_ret_hex,
                      :ifca_res_hex,
                      :buff_overflow,
                      :return_buff :return_ind);
```

Use the DSNTEJ6S installation verification program (IVP) to precompile, compile, link-edit, and bind the DSN8EP2 program. The DSNTEJ6S IVP also runs the following INSERT statement:

```
INSERT INTO SYSIBM.SYSPROCEDURES
VALUES ('DSN8EP2',
        ' ',
        ' ',
        'DSN8EP2',
        ' ',
        'DSN8STOR',
        'PLI',
```

```
        0,
        ' ',
          ,
        'N',
        ' ',
          ,
        'VARCHAR(4096) IN, INTEGER OUT, INTEGER OUT, INTEGER OUT, VARCHAR(409) OUT');
```

Use the DSNTEJ8S IVP job to precompile, compile, link-edit, and bind the DSN8ED1 program.

Note: A plan is required in this case because the requester (DSN8ED1) runs in an MVS environment. You do not have to include the package of the stored procedure in the requester plan.

Chapter 7. New Functions in DDCS and DB2 common server

In this chapter we discuss some new functions available in DDCS V2.3 and DB2 common server V2.1. We cover the following topics:

- Compound SQL
- Accounting information
- Tracing
- DRDA DBHEAP size
- Directory caching
- Hopping
- DDCS pre-fetch.

7.1 Compound SQL

With compound SQL, multiple SQL statements can be grouped into a single executable block. A compound SQL block can only be embedded in an AR. Compound SQL is not available in DB2 for MVS/ESA, but it is accepted from a DDCS workstation AR.

7.1.1 Syntax

There are two types of compound SQL, atomic and not atomic, as shown in Figure 86 on page 118 and described below.

ATOMIC compound SQL creates a savepoint at the beginning of a group of compound SQL statements and rolls back to this point in the event that an error is encountered during execution. The application receives a response from the database manager when all statements have completed successfully, or when one statement ends in error. If one statement ends in error, the entire block is considered to have ended in error, and any changes made to the database within the block are rolled back.

Note: ATOMIC compound SQL is not supported in a DDCS environment.

With not atomic compound SQL, the individual statements are a continuation of the current unit of work and can only be rolled back with this full unit of work. The application receives a response from the database manager when all statements have completed. All statements within a block are executed regardless of whether or not the preceding statement completed successfully. The group of statements can be rolled back only if the unit of work containing the NOT ATOMIC compound SQL is rolled back.

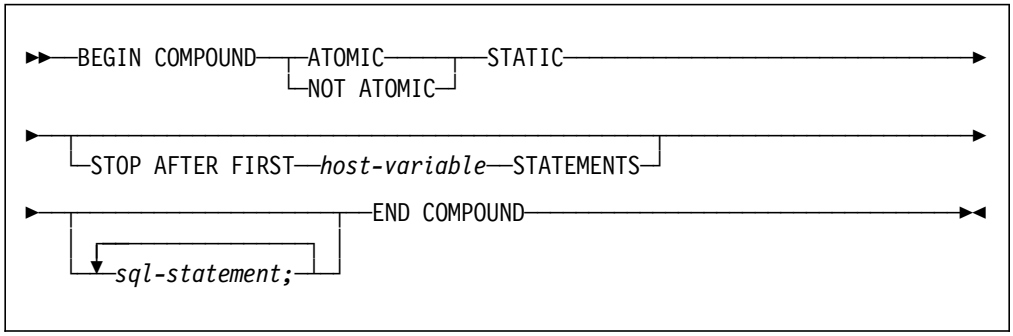


Figure 86. Compound SQL Syntax

The example below shows how to code the COMPOUND NOT ATOMIC SQL statement to insert four rows of data into a table. Note that there is no semicolon after the ATOMIC keyword, but there is a semicolon after each statement and after the END COMPOUND statement.

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC
    INSERT INTO STAFF (ID,NAME) VALUES ('10','USER1');
    INSERT INTO STAFF (ID,NAME) VALUES ('11','USER2');
    INSERT INTO STAFF (ID,NAME) VALUES ('12','USER3');
    INSERT INTO STAFF (ID,NAME) VALUES ('13','USER4');
    END COMPOUND;
```

See *DATABASE 2 SQL Reference for common server Version 2* for a full explanation of compound SQL syntax.

7.1.2 Rules

The following rules apply to compound SQL:

- The entire compound SQL statement construct is an executable statement that cannot be dynamically prepared.
- The compound SQL statement is not supported in REXX.
- No host language is allowed within a compound SQL statement; that is, no host language code is allowed between the substatements that make up the compound SQL statement.
- Only NOT ATOMIC compound SQL statements are accepted by DDCS V2.3.
- The compound SQL statement can be used by any AR to which DDCS V2.3 can connect.
- The compound SQL statement does not support the following SQL statements:
  - OPEN
  - CLOSE
  - CONNECT
  - DISCONNECT
  - RELEASE
  - SET CONNECTION

  - PREPARE
  - DESCRIBE
  - EXECUTE IMMEDIATE
  - FETCH
  - ROLLBACK
  - Compound SQL.
- Compound SQL statements cannot be nested.
- The user executing the compound SQL statement must have the appropriate authorization on all individual statements contained therein.
- A compound SQL statement can contain a COMMIT statement only if it is the last substatement of the group. If the COMMIT statement is the last substatement, it is always executed even if the STOP AFTER FIRST parameter specifies stopping before reaching the COMMIT statement. For example, if there are 100 statements, the last being the COMMIT, and STOP AFTER 50 is specified, the COMMIT statement is executed as the fifty-first statement.

7.1.3 Benefits

Compound SQL reduces database manager overhead because multiple statements are executed in a block.

For remote clients, compound SQL reduces the number of requests that have to be transmitted across the network. Because the information required for execution is passed to where it is executed as one group, rather than as seperate pieces of information for each SQL statement, network overhead is reduced and response time is improved.

An application that does batch inserts into a remote database would benefit from the use of NOT ATOMIC compound SQL. Rather than inserting each row of a table one by one and bearing the overhead associated with sending and receiving for each row, the application could group 100 or more inserts into a single block such that they would be transmitted in a continuous stream. This kind of transmission would reduce execution time, particularly in cases where communications create a performance bottleneck.

7.1.4 Debugging

If one more or errors occur, only one SQLCA is returned for the entire compound SQL.

The SQLCODE and SQLSTATE normally reflect the result for the last substatement. An exception is an SQLCODE +100 (no data found), which takes precedence over any other warning.

The SQLWARN indicators are an accumulation of the indicators set for all substatements.

The SQLERRMC, in a NOT ATOMIC compound SQL, contains information about up to a maximum of the last seven errors. The SQLERRMC is formatted as follows:

nnnXsssccccc

where:

nnn is the total number of statements that produced errors.

X is the token separator x'FF'.

sss is the ordinal position of the statement in error.

ccccc is the SQLSTATE of the error.

The second SQLERRD field contains the number of statements that failed (only negative SQLCODES).

The third SQLERRD field is an accumulation of the number of rows affected by all substatements.

The fourth SQLERRD field is a count of the number of successful substatements.

The fifth SQLERRD field is an accumulation of the number of rows updated or deleted due to referential integrity enforcement for each of the tables affected by the substatements.

7.1.5 Compound SQL with IMPORT

A new option has been added to the MODIFIED BY parameter of the IMPORT command to use NOT ATOMIC compound SQL. You can specify **COMPOUND=X**, where **X** is the number of insert statements to be performed at one time as part of a compound statement (see Figure 87). **X** must be between 1 and 100.

```
import from myfile.ixf of ixf modified by compound=100 insert into table
```

Figure 87. IMPORT Command

The COMPOUND=X option is not supported with the INSERT_UPDATE option of IMPORT (insert or update based on primary keys).

COMMITCOUNT should be a multiple of X.

Note: You must use IXF when importing to the host.

7.2 Accounting Information

DDCS V2.3 has a new function that enables the AR to send an accounting string to a DRDA AS. This function is implemented through the use of the PRDDTA parameter. The accounting string is sent to the DRDA AS at connection time.

Because the PRDDTA parameter is not architected in DRDA, there is no guarantee that your AS will recognize the data as accounting data.

Currently, only DB2 for MVS/ESA supports the receipt of accounting string information. Accounting string information sent to other DRDA ASs is ignored. DB2 for VM intends to provide support for this accounting data sent from DDCS V2.3.

The accounting string is part of the DB2 for MVS/ESA accounting trace and thus can go to SMF. It works with DB2 for MVS/ESA V3 and future versions. You have to add an APAR on DB2 for MVS/ESA V3 for the support of DSNDQMDA (APAR PN57323, PTF UN64166 on 9407). DB2PM V3.2 supports the accounting string, and you can view the data using the Long Accounting Trace, IFCID 3, and Online Monitor Requestor Correlation panels.

The accounting string sent in an application's connect request consists of a DDCS-generated prefix and user-defined suffix. The DDCS prefix can be 56 bytes and the user prefix can be up to 199 bytes, for a maximum length of 255 bytes. If a user prefix is longer than 199 bytes, it is truncated.

Table 13 shows the format of the DDCS prefix.

Table 13. DDCS-Generated Prefix Format

Field Name	Length	Description
acct_str_len	1	A hexadecimal value representing the length of the accounting string minus 1.
client_prdid	8	The product ID of the client's DB2 CAE software. For example, SQL02010 for DB2 CAE V2.
client_platform	18	The platform the client is on, for example, OS/2, Windows, DOS.
client_appl_name	20	The first 20 characters of the user's application name.
client_authid	8	The authid of the user's application.
suffix_len	1	A hexadecimal value representing the length of the user-supplied suffix.

You can specify the user-defined suffix in three ways.

1. Use the set accounting string API: SQLESACT().

   This API is called before the application connects to a database. You do not have to call this API again unless you want to use a new accounting string for future connect requests.

   You can write or modify your applications to use this new API to set the accounting string. Existing applications that are not rewritten to use this new API must use the environment variable or configuration parameter methods described below.

   We recommend that you use the set accounting string API method. See the *DATABASE 2 API Reference for common servers* for details on using this API.

2. Use the DB2ACCOUNT environment variable.

If the API was not called before the first *connect*, the DB2ACCOUNT environment variable is read. This value remains in effect until the end of the application or background CLP process.

3. Use the configuration parameter *dft_account_str* at the DDCS workstation.

   The value of the DFT_ACCOUNT_STR configuration parameter is the default accounting string. It is selected if the DB2ACCOUNT environment variable and API were not set. You have only one default accounting string per instance (defined in the database manager configuration file).

   If no value exists in the DFT_ACCOUNT_STR parameter, a null string is set for the user-defined suffix.

The accounting string data is converted to CCSID 500 before it is sent to the DRDA AS. For this reason and to ensure that the accounting string is converted correctly when it is transmitted to the DRDA server, you should use only the characters A to Z, 0 to 9, and the underscore (_).

If you initially want to do the accounting only at an instance level, make sure there are no values in SQLESACT() API, and that no values have been specified for the DB2ACCOUNT environment variable. Then update the database manager configuration file, setting a value for the DFT_ACCOUNT_STR.

If at a later time you want to determine which user is the most expensive, you can insert values in your DB2ACCOUNT environment variable or use SQLESACT() API. The DB2 for MVS/ESA server will then have your user's accounting DFT_ACCOUNT_STR in its accounting reports.

The following are examples of accounting strings:

```
X"3C"SQL020100S/2              cheque              SMITH  X"05"
DEPT1

X"3C"SQL020100S/2              cheque              SMITH  X"05"
```

In the first example, the user-defined suffix is DEPT1; in the second example, it is a null string.

7.3 Tracing

You can use the DDCSTRC command to trace the information exchanged between the DDCS workstation and the DRDA AS. The trace can help you determine the origin of a particular problem. The DDCSTRC executable is found in the SQLLIB\BIN directory.

The DDCS trace output shows:

- The first DDM command or object in a buffer, labeled as DSS TYPE.
- Data sent to the DRDA AS, labeled as SEND BUFFER.
- Data received from the DRDA AS, labeled as RECEIVE BUFFER.
- Data received from the DRDA AS that contains SQLCA information, labeled as SQLCA.

Figure 88 on page 123 shows an example of the DDCSTRC command, where:

on Turns on the DRDA application requester trace events

-r Traces DRDA data received from the DRDA AS

-s Traces DRDA data sent from the DRDA AS

-c Traces the SQLCA received from the DRDA AS

-t=ddcstrc.dmp Is the destination trace file

```
ddcstrc on -r -s -c -t=ddcstrc.dmp
```

Figure 88. DDCS Trace Command

7.4 DRDA DBHEAP Size

The DBHEAP is a new database manager configuration parameter that indicates the number of pages (4K) allocated for the memory used by the DRDA AS or the DRDA AR (DDCS).

There is one DBHEAP parameter for each database. The database manager uses the parameter on behalf of all CONNECT commands (applications or CLP) to a database. The amount of memory allocated for the heap contains control block information for tables, indexes, and tablespaces as well as space for the log buffer (LOGBUFSZ parameter). Therefore, the size of the heap depends on the following:

- Number of cursors opened
- Number of input variables
- Number of items in the select list
- Length of bound SQL statements

The control blocks allocated are freed only at disconnect from the database.

The default values depend on the operating system and whether the database server has remote clients. The default values are:

- UNIX: 1200 pages
- OS/2 database server with local and remote clients: 600 pages
- OS/2 database server with local clients: 300 pages.

We recommend not changing the default values. However when an application receives an error indicating that there is not enough storage available in the database heap, you must increase the DBHEAP parameter value. When setting this parameter, consider the value of LOGBUFSZ parameter because the log buffer is allocated from the database heap. Note that the DBHEAP parameter is allocated from the same shared memory as the buffer pool (BUFFPAGE parameter), lock list (LOCKLIST parameter), and utility heap (UTIL_HEAP_SZ parameter).

7.5 Directory Caching

To improve CONNECT performance, a new configuration option has been added to allow the DB2 common server directories to be cached in memory. When this option is set, a shared directory cache is built during DB2 initialization, and a private application directory cache is built when an application issues its first connect. Each cache provides an image of the system database directory, the DCS directory, and the node directory.

If the directory information cannot be found in the directory cache, the directory files on disk are searched.

The use of the directory cache reduces CONNECT costs by eliminating directory file I/O and minimizing directory searches required to retrieve directory information.

The default setting for directory caching is turned on. To update the directory caching setting, use the database director (see Figure 89) or run one of the following commands:

Updating Directory Caching from the Command Line

```
db2 update database manager configuration using dir_cache yes
db2 update database manager configuration using dir_cache no
```

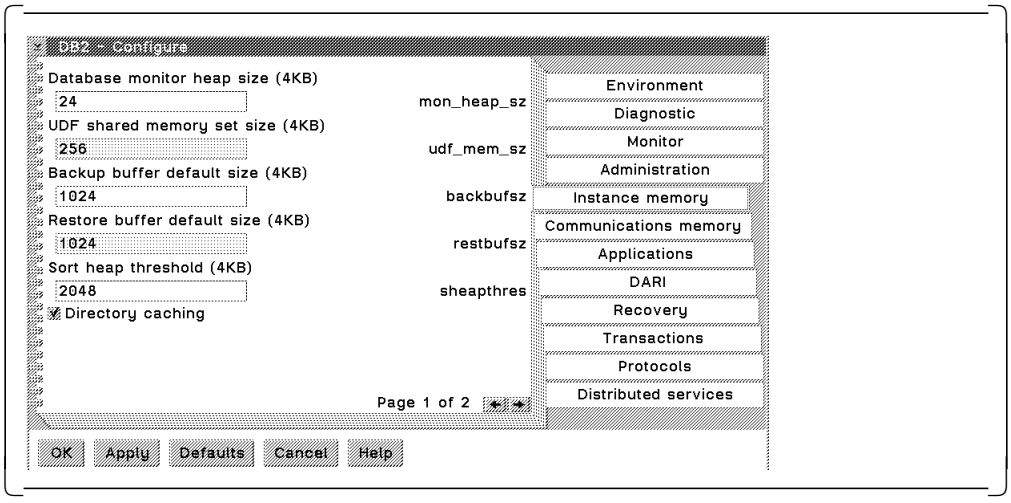


Figure 89. Updating Directory Caching

To refresh the directory cache used in a CLP session, issue a `db2 terminate` command.

Note: In a setup and/or development environment, we suggest that you disable directory caching, so that you do not have to continually stop and restart DB2.

7.6 Hopping

Hopping is a mechanism for accessing data from a remote database by using another database. The requester site is not aware that the data is not in the database specified in the CONNECT command. Hopping can occur only once. The DRDA protocol cannot be used in the “hop”; that is, the DB2 private protocol must be used.

Figure 90 depicts hopping in the DB2 common server environment, and Figure 91 on page 126 depicts hopping in the DB2 for MVS/ESA environment

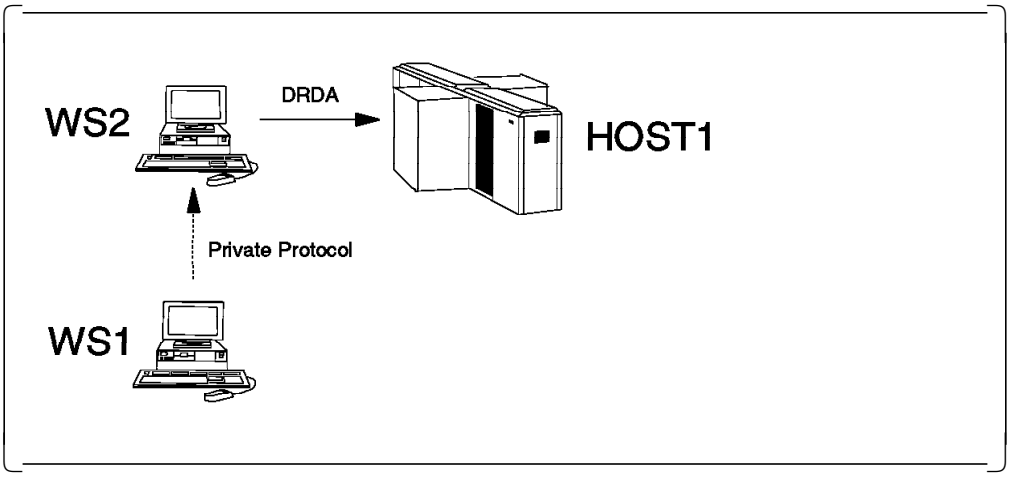


Figure 90. Hopping in the DB2 common server Environment

To implement hopping in the DB2 common server environment, define the connection from the DB2 common server intermediate workstation (WS2) to the host (HOST1) in the usual manner and catalog a database that references the host database. Then, on the workstation requester (WS1), catalog a database that references the database you just cataloged on WS2.

Once you issue connect statements to the local database, the statements are routed to WS2 through the hopping technique, and you are then connected to the remote host.

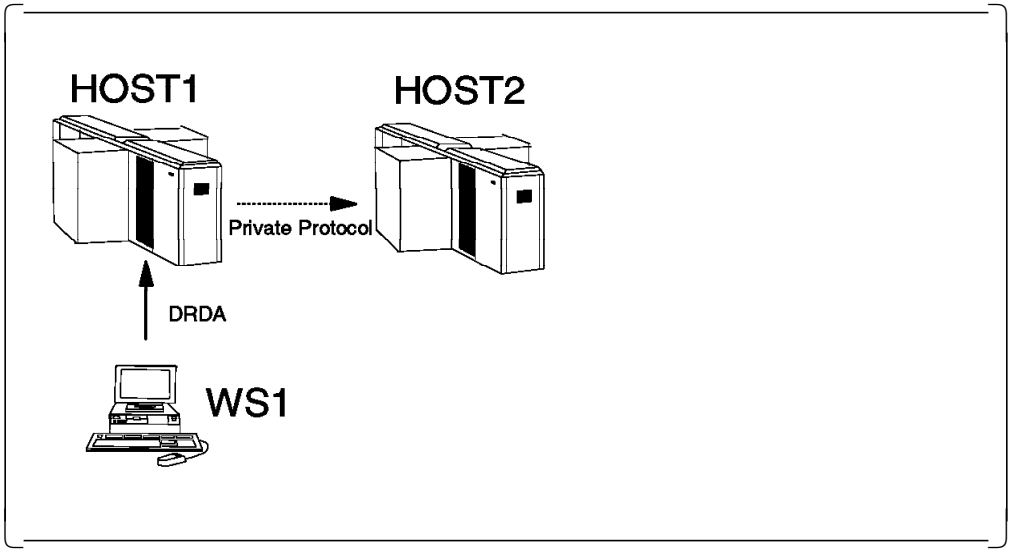


Figure 91. Hopping in the DB2 for MVS/ESA Environment

To implement hopping in the DB2 for MVS/ESA environment, simply define the connections from the DB2 common server (WS1) to the host database in the usual manner. Then in the intermediate DB2 for MVS/ESA AS (HOST1), define an ALIAS for the remote table. For example:

```
CREATE ALIAS MYTABLE FOR HOST2.EDWARDSC.SAMPTAB
```

Note: Hopping is not supported in the workstation environment for connections from the host to the workstation.

7.6.1 Pre-Fetch

Pre-fetch is designed to improve the response times for queries that return large result sets. It combines the blocking technique used in previous version of DDCS, plus DDCS′s ability to fetch rows of data for an open cursor before the application actually requests the data.

Pre-fetch is performed automatically and transparently every time you issue an SQL OPEN or FETCH statement. Basically, when you issue an SQL OPEN statement, DDCS gets a block of data. If the query still has more data on the server side, DDCS asynchronously issues another DRDA FETCH to get the next block of data. The first block is processed by the subsequent SQL FETCH statement. The next time DDCS has to send out another DRDA FETCH request for the next block, chances are the next block is already in the workstation. At this time DDCS issues another asynchronous DRDA FETCH to get a subsequent block.

Pre-fetch differs from DB2 for MVS/ESA continuous block fetching in that DDCS does not need a separate LU 6.2 session and does not tie up extra VTAM buffers along the way. Continuous block fetching keeps pumping the data, which is buffered and paused by VTAM, from the DRDA AS to the AR. DDCS only requests one extra block at a time. In practice an extra block at a time is sufficient for most machines and usage. Any more than that could tie up the machine cycle and buffers because the application is too slow to fetch all of the data.

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

X

ITSO Redbook Evaluation

International Technical Support Organization
DB2 for MVS Connections with AIX and OS/2
November 1995

Publication No. SG24-4558-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **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) Are you an employee of IBM or its subsidiaries: Yes____ No____

b) Do you work in the USA? Yes____ No____

c) Was this redbook published in time for your needs? Yes____ No____

d) Did this redbook meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this redbook?

_____

_____

What other redbooks would you like to see published?

_____

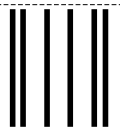
Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

Name	Address
Company or Organization	
Phone No.	

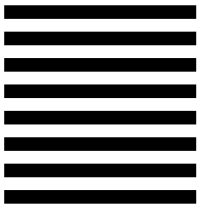
ITSO Redbook Evaluation
SG24-4558-00

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/E2
650 Harry Road
San Jose, CA
USA 95120-6099

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.