DB2 Server for VSE

System Administration

Version 7 Release 5

SC09-2981-03

IBM

DB2 Server for VSE

System Administration

Version 7 Release 5

SC09-2981-03

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 423.

Contents

About This Manual

This manual describes how to carry out system planning and administration tasks for DB2 Server for VSE.

Note: If your installation is using VSE guest-sharing to access a database manager on a VM operating system, you can use this manual to carry out tasks that involve operating the database manager or improving performance on VSE. However, for a complete description of VM administrative tasks, including those that involve the database virtual machine, you will need the *DB2 Server for VM System Administration* manual.

The following tasks are described here:

- Installation
- Migration
- Operation
- Management of resources (including security)
- Modification of facilities (including national language support).

Organization of This Manual

- "Summary of Changes" on page xvii lists the changes made to the product since Version 7 Release 4.
- Chapter 1, "Planning for Installation," on page 1 summarizes the software, hardware, and storage requirements for installing the database manager. It does not describe the actual installation procedure. For information on that, see the *DB2 Server for VSE Program Directory*.
- Chapter 2, "Planning for Database Generation," on page 13 describes how to set up your initial database, including specifying parameters to define the logical and physical limits for its capacity and setting its initial DASD allocations.
- Chapter 3, "Planning for Database Migration," on page 39 explains the planning you must do before migrating a database from a previous release of the database manager to the Version 7 Release 5 level. For the actual migration steps, see the *DB2 Server for VSE Program Directory*.
- Chapter 4, "Planning for Operation of the Database Manager," on page 47 explains how to choose appropriate startup parameters which will determine the operational characteristics of the application server when it is started by the DB2 Server for VSE operator.

  Note: Starting, operating, and stopping the application server are also discussed in the *DB2 Server for VSE & VM Operation* manual.
- Chapter 5, "Operating the Online Support ," on page 81 explains how to enable VSE guest users to access the application server on a VM/ESA operating system, and how to operate the online support for CICS/VSE® transactions.

  Note: These subjects are also discussed in the *DB2 Server for VSE & VM Operation* manual.
- Chapter 6, "Maintaining Database Security," on page 119 discusses how to control access to the application server.

- Chapter 7, “Managing Database Storage,” on page 123 explains how to manage the disk storage allocated to the database, including adding (or defining) dbspaces, defining storage pools, adding dbextents to storage pools, and managing storage pools.
- Chapter 8, “Making Backups and Recovering from Failures,” on page 143 describes facilities provided for recovery from system failures and DASD failures; how to back up your database; and how to recover from different types of failures.
- Chapter 9, “Special Topics in Recovery Design,” on page 167 discusses dual logging and switching log modes.
- Chapter 10, “Using the Accounting Facility,” on page 187 describes the DB2 Server for VSE accounting facility, which tracks how database resources are consumed by users.
- Chapter 11, “Generating Additional Databases,” on page 211 describes how to add databases to your system.
- Chapter 12, “Choosing a National Language and Defining Character Sets,” on page 231 contains information on national language character set and coded character set identifier (CCSID) support, as well as how to provide HELP text and messages in languages supported by the database manager.
- Chapter 13, “Creating Installation Exits,” on page 263 describes the types of installation exits that you can code to customize the database manager:
  - Accounting exits, to customize account information
  - Date and time exits, to create your own date or time format if the IBM-supplied formats do not fit your requirements
  - TRANSPROC exits, to carry out DBCS conversions
  - Cancel exits, to replace the product-supplied cancel function when coding your own interactive program
  - Field Procedures, to change the sorting sequence by encoding and decoding data if the standard sorting sequence does not meet your requirements.
- Chapter 14, “Using a DRDA Environment,” on page 313 discusses using the database manager in a distributed environment.
- Chapter 15, “Using TCP/IP with DB2 Server for VSE,” on page 335 discusses using TCP/IP to access application servers.
- Appendix A, “Processor Storage Requirements,” on page 339 presents guidelines for estimating the processor requirements needed for running the database manager.
- Appendix B, “Estimating Database Storage,” on page 341 contains procedures for estimating the sizes of the database directory, public dbspaces, and the ISQL dbspace.
- Appendix C, “Maximum Values,” on page 357 contains the system and database maximums for the database manager.
- Appendix D, “Updating SYSTEM.SYSSTRINGS,” on page 359 details how to update this catalog table to support your own CCSID conversion.
- Appendix E, “Defining Your Own Character Set,” on page 363 describes how to create your own character set.
- Appendix F, “Macro List,” on page 379 lists the macros identified as programming interfaces for customers by the database management system.
- Appendix G, “Service and Maintenance Utilities,” on page 381 lists and describes service and maintenance utilities.
- Appendix H, “DRDA Considerations,” on page 385 discusses what you should consider in a distributed environment.

- Appendix I, “Incompatibilities Between Releases,” on page 389 describes the incompatibilities between releases.

A bibliography is provided at the back of the book.

Syntax Notation Conventions

Throughout this manual, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right and from top to bottom, following the path of the line.

  The ►►── symbol indicates the beginning of a statement or command.

  The ──► symbol indicates that the statement syntax is continued on the next line.

  The ►── symbol indicates that a statement is continued from the previous line.

  The ──►◄ symbol indicates the end of a statement.

  Diagrams of syntactical units that are not complete statements start with the ►── symbol and end with the ──► symbol.
- Some SQL statements, Interactive SQL (ISQL) commands, or database services utility (DBS Utility) commands can stand alone. For example:

```
►►──SAVE──────────────────────────────────────────────────────────►◄
```

  Others must be followed by one or more keywords or variables. For example:

```
►►──SET AUTOCOMMIT OFF────────────────────────────────────────────►◄
```

- Keywords may have parameters associated with them which represent user-supplied names or values. These names or values can be specified as either constants or as user-defined variables called *host_variables* (*host_variables* can only be used in programs).

```
►►──DROP SYNONYM──synonym─────────────────────────────────────────►◄
```

- Keywords appear in either uppercase (for example, SAVE) or mixed case (for example, CHARacter). All uppercase characters in keywords must be present; you can omit those in lowercase.
- Parameters appear in lowercase and in italics (for example, *synonym*).
- If such symbols as punctuation marks, parentheses, or arithmetic operators are shown, you must use them as indicated by the syntax diagram.
- All items (parameters and keywords) must be separated by one or more blanks.
- Required items appear on the same horizontal line (the main path). For example, the parameter *integer* is a required item in the following command:

```
►►──SHOW DBSPACE──integer──────────────────────────────────────────────►◄
```

This command might appear as:

```
SHOW DBSPACE 1
```

- Optional items appear below the main path. For example:

```
►►──CREATE──┬────────┬──INDEX──────────────────────────────────────────►◄
            └─UNIQUE─┘
```

This statement could appear as either:

```
CREATE INDEX
```

or

```
CREATE UNIQUE INDEX
```

- If you can choose from two or more items, they appear vertically in a stack.

  If you must choose one of the items, one item appears on the main path. For example:

```
►►──SHOW LOCK DBSPACE──┬─ALL─────┬─────────────────────────────────────►◄
                       └─integer─┘
```

Here, the command could be either:

```
SHOW LOCK DBSPACE ALL
```

or

```
SHOW LOCK DBSPACE 1
```

If choosing one of the items is optional, the entire stack appears below the main path. For example:

```
►►──BACKWARD──┬─────────┬──────────────────────────────────────────────►◄
              ├─integer─┤
              └─MAX─────┘
```

Here, the command could be:

```
BACKWARD
```

or

```
BACKWARD 2
```

or

```
BACKWARD MAX
```

- The repeat symbol indicates that an item can be repeated. For example:

```
          ┌──────┐
          ▼      │
►►──ERASE───name─┴──────────────────────────────────────────────►◄
```

  This statement could appear as:

  ```
  ERASE NAME1
  ```

  or

  ```
  ERASE NAME1 NAME2
  ```

  A repeat symbol above a stack indicates that you can make more than one choice from the stacked items, or repeat a choice. For example:

```
                 ┌─,──────────────────────┐
                 ▼                        │
►►──VALUES──(──┬─constant──────────┬──┴──)─────────────────────►◄
               ├─host_variable_list─┤
               ├─NULL───────────────┤
               └─special_register───┘
```

- If an item is above the main line, it represents a default, which means that it will be used if no other item is specified. In the following example, the ASC keyword appears above the line in a stack with DESC. If neither of these values is specified, the command would be processed with option ASC.

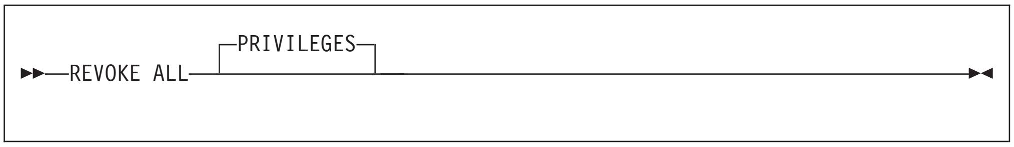
```
     ┌─ASC──┐
►►───┼──────┼───────────────────────────────────────────────────►◄
     └─DESC─┘
```

- When an optional keyword is followed on the same path by an optional default parameter, the default parameter is assumed if the keyword is not entered. However, if this keyword is entered, one of its associated optional parameters must also be specified.

  In the following example, if you enter the optional keyword PCTFREE =, you also have to specify one of its associated optional parameters. If you do not enter PCTFREE =, the database manager will set it to the default value of 10.

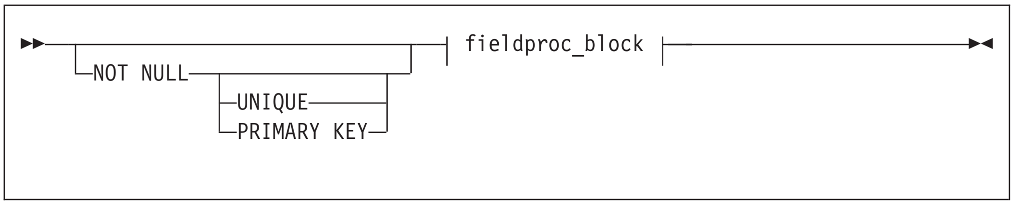
```
     ┌─PCTFREE = 10──────┐
►►───┼───────────────────┼──────────────────────────────────────►◄
     └─PCTFREE = integer─┘
```

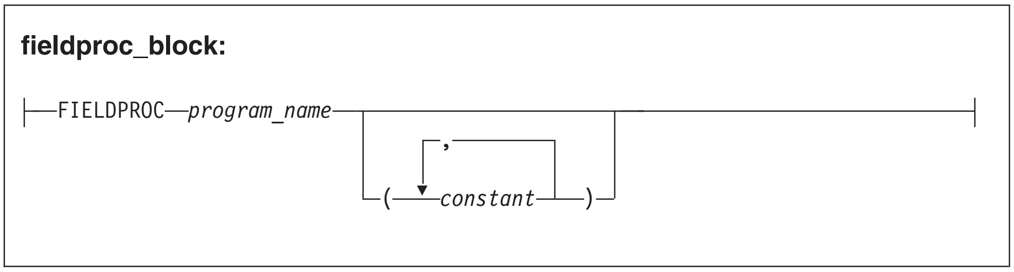
- Words that are only used for readability and have no effect on the execution of the statement are shown as a single uppercase default. For example:



Here, specifying either REVOKE ALL or REVOKE ALL PRIVILEGES means the same thing.

- Sometimes a single parameter represents a fragment of syntax that is expanded below. In the following example, **fieldproc_block** is such a fragment and it is expanded following the syntax diagram containing it.





SQL Reserved Words

The following words are reserved in the SQL language. They cannot be used in SQL statements except for their defined meaning in the SQL syntax or as host variables, preceded by a colon.

In particular, they cannot be used as names for tables, indexes, columns, views, or dbspaces unless they are enclosed in double quotation marks (").

ACQUIRE
ADD
ALL
ALTER
AND
ANY
AS
ASC
AVG

BETWEEN
BY

CALL
CHAR
CHARACTER
COLUMN
COMMENT
COMMIT
CONCAT
CONNECT
COUNT
CREATE
CURRENT

DBA
DBSPACE
DELETE
DESC
DISTINCT
DOUBLE
DROP

EXCLUSIVE
EXECUTE
EXISTS
EXPLAIN

FIELDPROC
FOR
FROM

GRANT
GRAPHIC
GROUP

HAVING

IDENTIFIED
IN
INDEX
INSERT
INTO
IS

LIKE
LOCK
LONG

MAX
MIN
MODE

NAMED
NHEADER
NOT
NULL

OF
ON
OPTION
OR
ORDER

PACKAGE
PAGE
PAGES
PCTFREE
PCTINDEX
PRIVATE
PRIVILEGES
PROGRAM
PUBLIC

RESOURCE
REVOKE
ROLLBACK
ROW
RUN

SCHEDULE
SELECT
SET
SHARE
SOME
STATISTICS
STORPOOL
SUM
SYNONYM

TABLE
TO

UNION
UNIQUE
UPDATE
USER

VALUES
VIEW

WHERE
WITH
WORK

Summary of Changes

This is a summary of the technical changes to the DB2 Server for VSE & VM database management system for this edition of the book. Several manuals are affected by some or all of the changes discussed here. For your convenience, the changes made in this edition are identified in the text by a vertical bar (|) in the left margin. This edition may also include minor corrections and editorial changes that are not identified.

This summary does not list incompatibilities between releases of the DB2 Server for VSE & VM product; see either the *DB2 Server for VSE & VM SQL Reference*, *DB2 Server for VM System Administration*, or the *DB2 Server for VSE System Administration* manuals for a discussion of incompatibilities.

Summary of Changes for DB2 Version 7 Release 5

Version 7 Release 5 of the DB2 Server for VSE & VM database management system is intended to run on the Z/VM Version 5 Release 2 or later environment and on the Z/VSE($^{®}$) Version 3 Release 1 or later environment.

Enhancements, New Functions, and New Capabilities

The following have been added to DB2 Version 7 Release 5:

Explain Option on DBSU REBIND PACKAGE Command

This new functionality allows the EXPLAIN(YES/NO) option on REBIND PACKAGE command. If EXPLAIN(YES) is issued, then all four update tables (structure, plan, cost, reference) will be updated. If EXPLAIN(NO) is issued, then none of the four update tables will be updated.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE & VM Database Services Utility*
- *DB2 Server for VSE & VM Performance Tuning Handbook*
- *DB2 Server for VSE & VM Quick Reference*
- *DB2 Server for VSE & VM SQL Reference*

For Fetch only

This new functionality accepts the ″FOR FETCH ONLY″ clause after a cursor select statement. It causes a cursor to become read-only (no UPDATEs or DELETEs are permitted using this cursor). If a read-only cursor is referenced in an UPDATE or DELETE statement, SQLCODE -510 will be issued and the statement is not processed. In addition, under the SBLOCK preprocessor option, ″FOR FETCH ONLY″ forces blocking to be used on the read-only cursor regardless of whether there is a COMMIT. If there is no ″FOR FETCH ONLY″ clause, under SBLOCK, blocking would only be done if a COMMIT was absent.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VM Messages and Codes*
- *DB2 Server for VSE & VM Application Programming*
- *DB2 Server for VSE & VM Performance Tuning Handbook*
- *DB2 Server for VSE & VM Quick Reference*

- *DB2 Server for VSE & VM SQL Reference*

Application Message Formatter

This functionality provides an Application Programming Interface (API) that retrieves the descriptive text for an SQLCODE, given an SQLCA input parameter. The API will be available for Assembly, COBOL, C, PL/I and FORTRAN.

In DB2 for VM and DB2 for VSE Online, the user may specify the language of the returned text. The languages supported by DB2 for VSE/VM are American English (AMENG), uppercase English (UCENG), German (GER), French (FRANC) and Japanese (KANJI). VSE Batch does not support switching to another language. Therefore the default will be used regardless of the user's specification. The values of SQLCODE, SQLSTATE, SQLERRD1 and SQLERRD2 will be automatically appended to the returned text. The user may also specify to have the entire SQLCA included. If the SQLCODE could not be found in the repository, the entire SQLCA will be returned in the buffer.

If the SQLCA was set by another product (such as DB2 UBD), the descriptive text is retrieved if the SQLCODE exists in the DB2 for VM/VSE repositories. However, the token substitutions may not be correct.

For more information, see *DB2 Server for VSE & VM Application Programming*.

Convert buffer read/write to compiler macro

The DRDA code has over 100 small modules. Each call to an external module has a certain amount of overhead associated with it. Certain modules are called very frequently and this can add up to a significant amount of time. This functionality improves the performance by converting few modules to macros or internal procedures, to reduce this overhead.

Modify Build Tree Creation

This functionality modifies Build Tree creation used by DRDA parsing and generation. It is built in such a way that every code point that is used to search through the tree must be converted to a different format before the search can be done. If modified build tree was created with the converted point, then the code point would not have to be converted every time the tree must be searched. This improves the performance of the DRDA code path length with the minimal search.

Split code point search routines

When parsing a data stream within each parser action routine, a binary search is done to find the specific code point. Some action specific routines are quite large, so the binary search can be long. Splitting and spreading the code point evenly among other modules would reduce the overheads and improves the performance of the DRDA code path length.

DRDA Multi-Row Insert

Multi Row insert is a means of caching homogenous insert statements and sending them as a block to the server for processing. This reduces the overhead of sending a large number of singular inserts and receiving as many responses.

Buffering of homogenous inserts eliminates the need to send an SQL statement to the DB2 server every time an insert is made, thereby improving performance over DRDA.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE & VM Application Programming*

- *DB2 Server for VSE & VM Database Administration*
- *DB2 Server for VM System Administration*
- *DB2 Server for VSE & VM Performance Tuning Handbook*
- *DB2 Server for VSE & VM Quick Reference*
- *DB2 Server for VSE & VM SQL Reference*

Connection Pooling for DRDA TCP/IP in Online Resource Adapter

Connection pooling is a technique that allows multiple users to share a cached set of pre-established connections that provide access to a database. Establishing a connection between a user and a server takes a sizeable time. Users who have validated their entry to a database once need not establish a connection every time a request is submitted. Instead, they can use a pre-established connection from a pool of such connections and get their results much faster.

From the user's point of view, there is a considerable improvement in response time after this line item is implemented.

For more information, see the following documentation on DB2 Server for VSE & VM:

- *DB2 Server for VSE System Administration*
- *DB2 Server for VSE & VM Application Programming*
- *DB2 Server for VSE & VM Operation*
- *DB2 Server for VSE & VM Performance Tuning Handbook*

IBM DB2 Server for VSE, Client Edition

This feature allows the customer the flexibility to install and use only the client (run-time support) component of DB2 Server for VSE without the requirement to buy and install the server component during the installation process of DB2 server for VSE product. The client-only installation enables customers to reduce the total cost of ownership when they have their databases residing on a non-local platform (like VM, z/OS, LUW) and have a large number of their DB2 applications on VSE (like ISQL on CICS, DBSU on VSE, other online/batch applications on VSE).

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE System Administration*
- *DB2 Server for VSE Program Directory*

IBM DB2 Server for VM, Client Edition

This feature allows the customer the flexibility to install and use only the client (run-time support) component of DB2 Server for VM without the requirement to buy and install the server component during the installation process of DB2 server for VM product. The client-only installation enables our customers to reduce the total cost of ownership when they have their databases residing on a non-local platform (like VM, z/OS, LUW) and have a large number of their DB2 applications on VM (like ISQL, DBSU, other user applications on VM).

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VM System Administration*
- *DB2 Server for VM Program Directory*

Handling Commit Responses from DB2 UDB Stored Procedures

This feature will allow DB2 Resource Manager on VSE/VM to accept and process results of a stored procedure running in a UDB server with a COMMIT statement in the stored procedure.

Currently, DB2 for VM/VSE client does not handle responses from ’COMMIT’ statements coded in DB2 UDB stored procedures. Implementation of this feature will enable handling responses of COMMIT statements in DB2 UDB stored procedures and thus allow users to have COMMIT statements in their stored procedures, while using DB2 for VM/VSE client.

COMMIT statements, however, are not allowed in stored procedures on the DB2 Server for VM/VSE.

For more information, see *DB2 Server for VSE & VM Application Programming*.

Make on-line programs AMODE 31 RMODE ANY

This feature converts DB2 server for VSE online program which presently operate under 24 bit addressing mode from AMODE 24, to AMODE 31 RMODE ANY. Presently, all the online programs are loaded below 16M line. Implementation of this line item ensures that all the online program will be loaded above the 16M line, which results in more virtual storage below the line, which can be utilized by other applications.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE System Administration*
- *DB2 Server for VSE Program Directory*

Provide BIND File Support in VM and in VSE Batch Environments

This feature provides the facility of binding packages across servers. The process of binding is achieved by dividing the program preparation method into two steps. The first step does the precompilation of the embedded SQL programs with the prep parameter ’BIND’. Invocation of VSE/VM preprocessor creates a ’bindfile’. The bindfile can be bound against any DB2 server using VSE/VM binder. During this process, the access path is generated, SQL statements are verified, authorization checks are performed, and package on the target server is created. This line item eliminates the need of re-prepping the source code or porting of packages across DB2 servers.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 REXX SQL for VM/ESA Installation and Reference*
- *DB2 Server for VM Messages and Codes*
- *DB2 Server for VSE & VM Application Programming*
- *DB2 Server for VSE & VM Database Administration*
- *DB2 Server for VM Program Directory*
- *DB2 Server for VSE Program Directory*

Convert TCP/IP LE/C interface to EZASMI API

The feature of converting TCP/IP LE/C interface to EZASMI API intends to replace the current LE/C interface and implement the EZA Assembler Interface (EZASMI)to enhance performance in DB2 Client/Server for VSE over DRDA. Currently, either LE/C interface or CSI Assembler Interface is used for TCP/IP functions. The EZASMI interface makes the code all Assembler.

For more information, see *DB2 Server for VSE Program Directory*

Chapter 1. Planning for Installation

Before installing the database manager, you must:

- Determine which of the usage environments will be appropriate for your processing requirements
- Understand a typical DB2 Server for VSE configuration
- Have the appropriate prerequisite programs
- Determine your virtual storage requirements
- Determine your hardware requirements
- Determine the DBNAME directory requirements
- Set up the DB2 key, if you are using z/VSE 3.1

Usage Environments

To determine what software and hardware you need, you must first categorize your planned use of the database manager as one or more of the following:

- Batch application processing
- Online (CICS) transaction processing
- Interactive application development
- Query or report writing

Each of these environments is described in detail below, including the DB2 Server for VSE functions supported, and options that are either recommended or required for the associated program products.

Batch Application Processing

Batch processing, as shown in Figure 1, is used for submitting jobs to run in a non-interactive mode. A job is submitted using job control (JCL) with or without the VSE/ICCF facilities. The only requirement beyond the base prerequisites is one of the supported programming languages.

Batch facilities are useful for job-preparation tasks like preprocessing and compiling host language programs, or for running applications. Application programs that do not require end-user access can be run in batch. Batch processing places minimal demands on system resources (real storage and processor power).

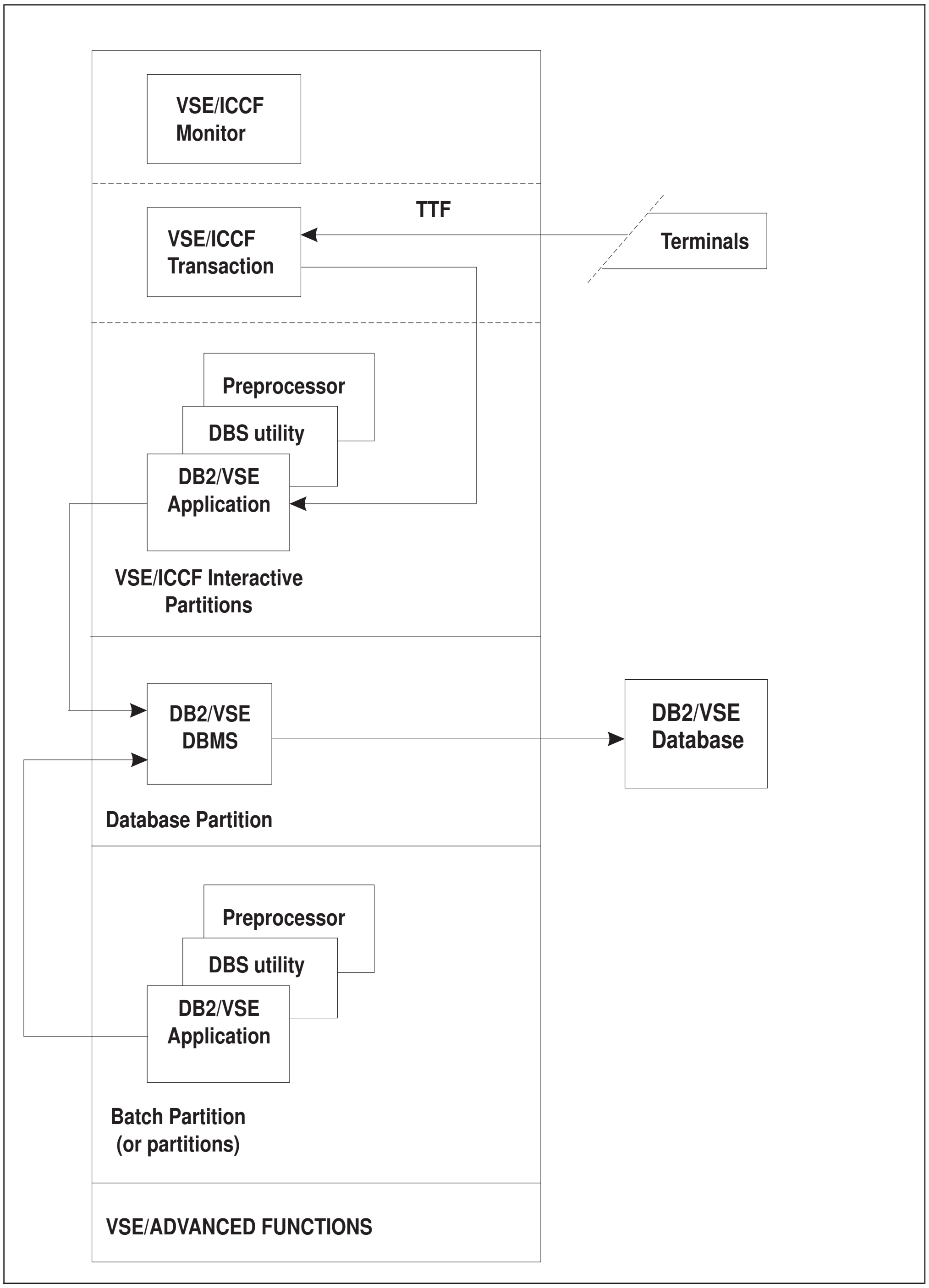


Figure 1. Batch Configuration

Online (CICS) Transaction Processing

Online transaction processing, as shown in Figure 2 on page 4, requires installation of the online support, and of CICS/VSE, or an equivalent product to support double-byte character set (DBCS) characters and to provide the terminal management and transaction-processing support. Online programs can be written in any of the supported programming languages. You can install ISQL, but its use is limited to data administration functions. Consider this environment for preplanned business applications where end user access to the system is managed through CICS/VSE transactions programmed for specific end user tasks.

For this environment, configure the system as follows:

- CICS/VSE options:
  - Dynamic transaction backout program (DBP): required for proper coordination and recovery with the database manager.
  - Exec level support: required to support transaction access to the database manager.
  - User exit interface: also required for transaction access to the database manager.
  - CICS/VSE monitoring facility: optional but recommended. If it is used, the database manager participates in the monitoring by providing performance class information.
  - CICS/VSE restart resynchronization: required if you plan to access databases from the CICS online environment.
- VSE/Power facility: required for the system printer or remote workstation printer report writing support in ISQL (which runs as a CICS/VSE transaction); not required for report writing to a CICS/VSE terminal printer. This is the only facility to provide multiple copy capability.

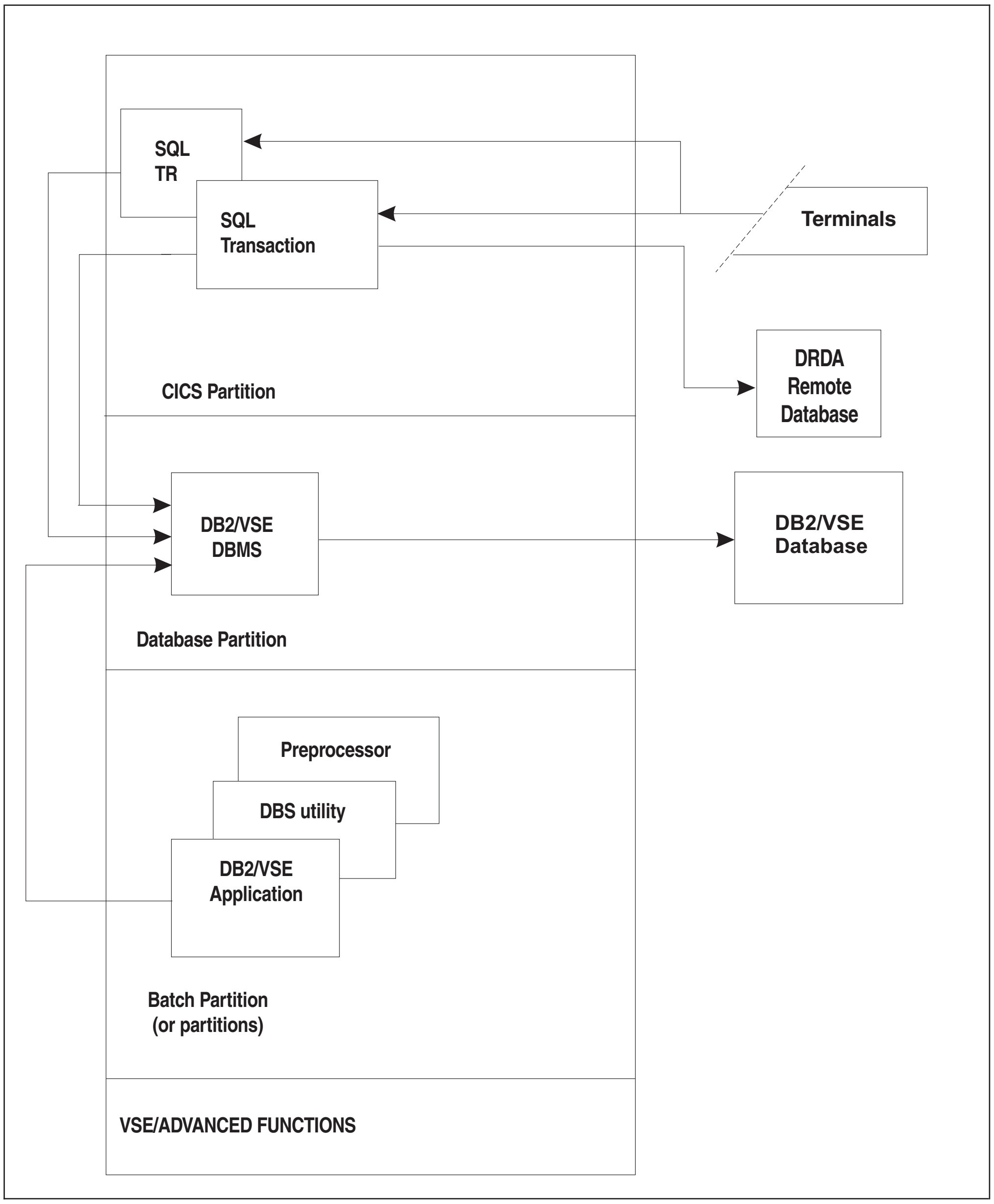


Figure 2. Online Transaction Processing Configuration

Interactive Application Development

An application development environment, as shown in Figure 3 on page 5, involves a large amount of data design, application coding, and testing. Such activities typically involve less SQL activity in the form of data definition, catalog queries, and program preprocessing. Correspondingly there is greater demand for real storage and processor resources than that demanded by application or transaction processing.

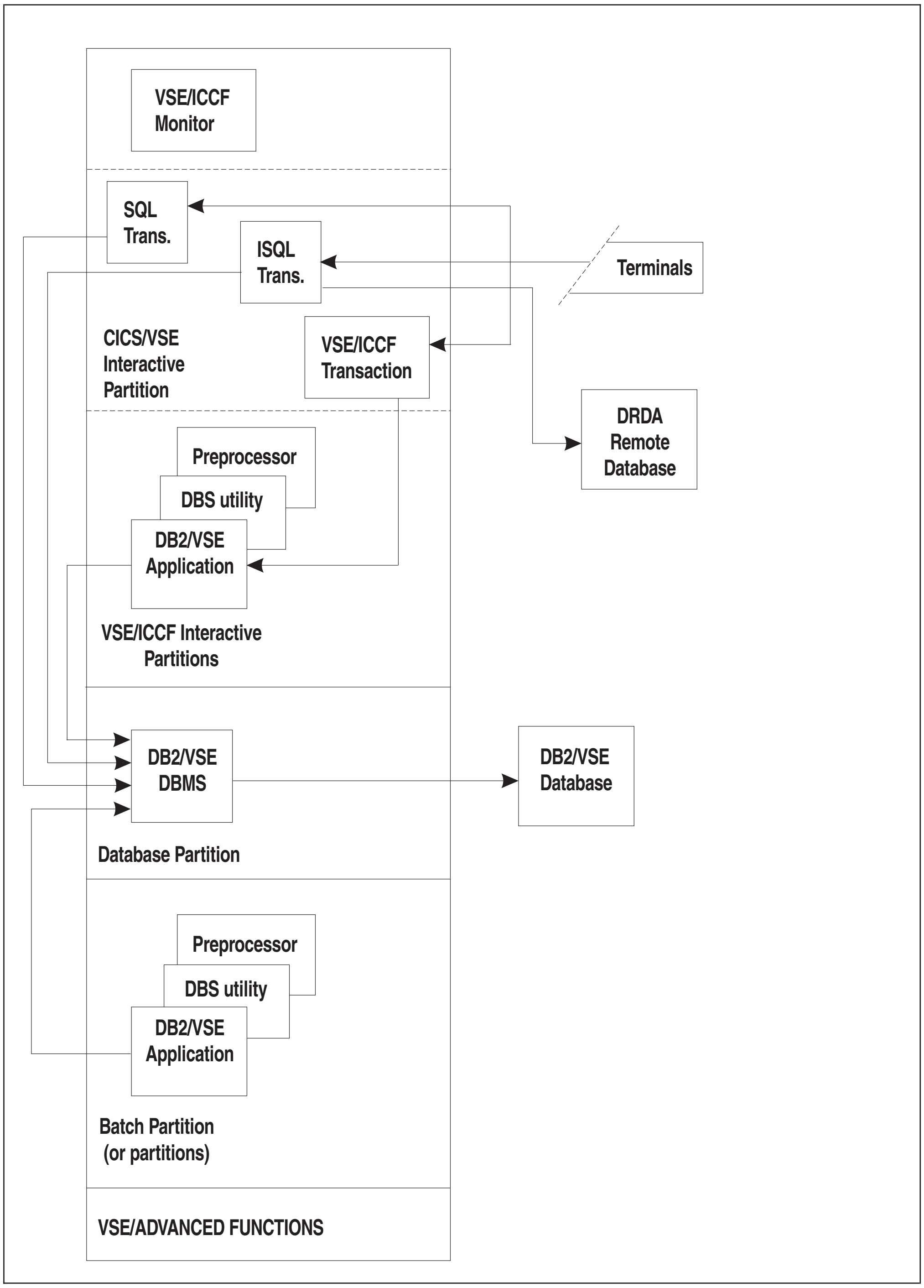


Figure 3. Interactive Application Development Configuration

The configuration requirements are the same as those described for online transaction processing.

Query/Report Writing

The query/report writing environment, as shown in Figure 4 on page 6, supports dynamic SQL query and report writing by end users. This environment places a relatively high demand on system resources, because user requests must be dynamically interpreted, the number of requests is not limited, and sorting may be very frequent.

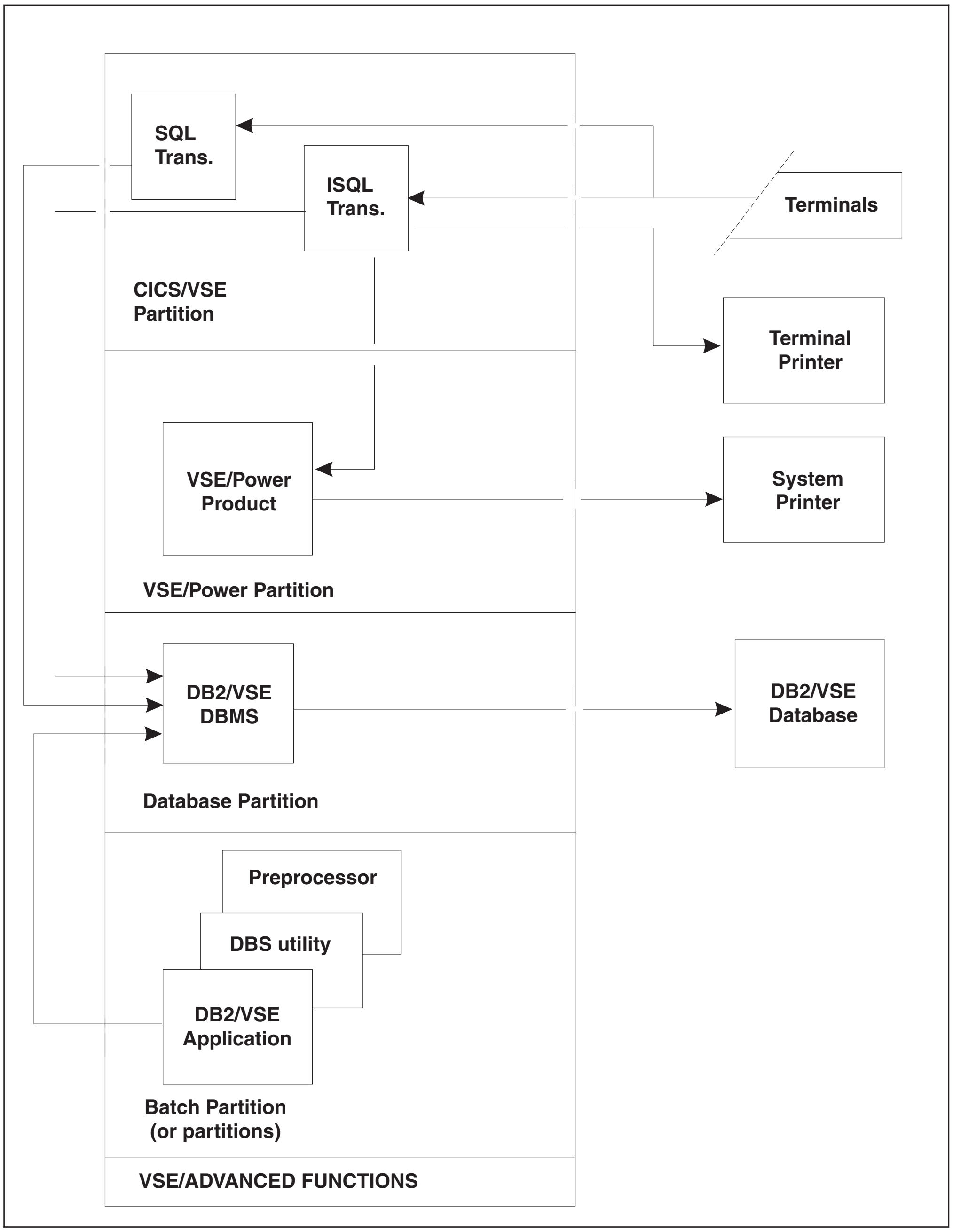


Figure 4. Query/Report Writing Configuration

The configuration requirements are the same as those described for the online transaction processing environment.

Components of the Relational Database Management System

Figure 5 on page 7 depicts a typical configuration with one database, one batch partition user, and a CICS® partition with several interactive users.

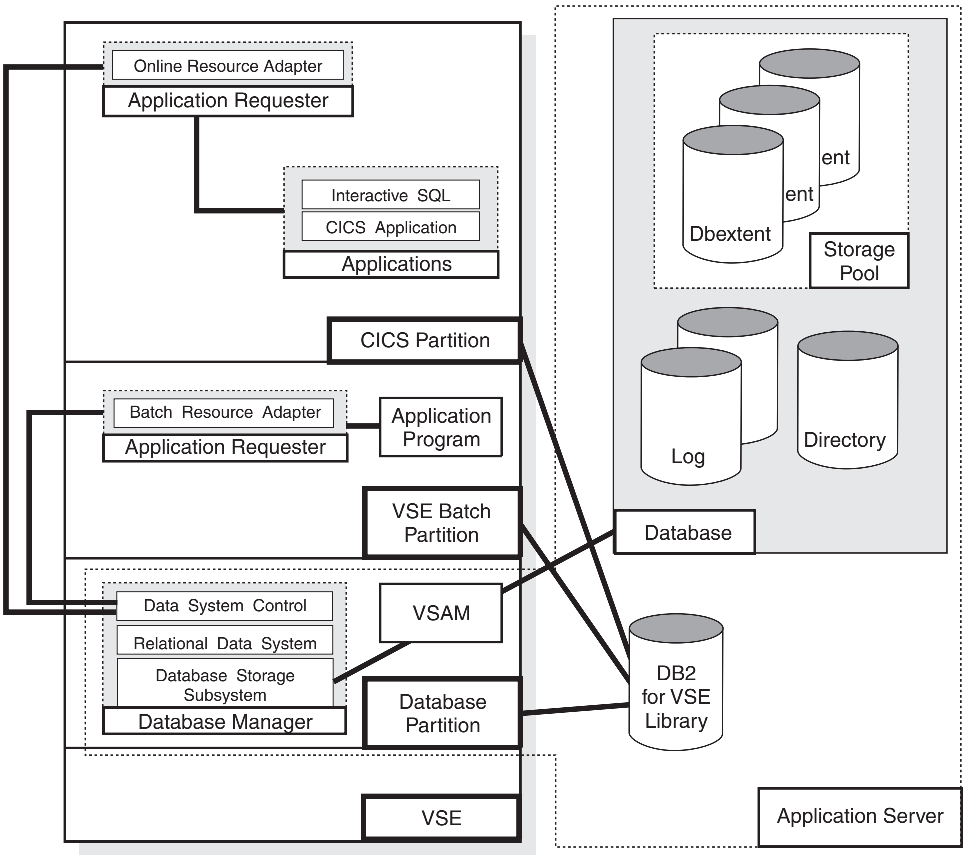


Figure 5. Basic Components of the RDBMS in VSE

The **database** is composed of :
- A collection of data contained in one or more *storage pools*, each of which in turn is composed of one or more *database extents (dbextents).*
- A *directory* that identifies data locations in the storage pools. There is only one directory per database.
- A *log* that contains a record of operations performed on the database. A database can have one, two, or four logs.

The **database manager** is the program that provides access to the data in the database. It is loaded into the database partition from the DB2 Server for VSE library.

The **application server** is the facility that responds to requests for information from and updates to the database. It is composed of the database and the database manager.

The **application requester** is the facility that transforms a request from an application into a form suitable for communication with an application server.

Prerequisite Programs

This section summarizes the program products required or recommended for the DB2 Server for VSE functions and environments available with this release. Unless otherwise stated, the database manager works with all subsequent versions, releases, and modification levels of the products listed in this section as well as with equivalent non-IBM products.

When installing this product on a VSE system, you require an environment provided by VSE/Enterprise Systems Architecture (VSE/ESA) Version 2 Release 3 Modification 1 or later. DB2 Server for VSE requires VSE/VSAM which is included with this operating system.

Table 1 summarizes the components needed for the usage environments, as well as the languages supported by each environment.

Table 1. VSE Environments

	DB2/VSE Components [1]			**Supported Languages**					
DB2/VSE Environments	**Base**	**ORA**	**ISQL**	**PL/I**	**COBOL**	**COBOL II**	**FOR-TRAN**	**C**	**ASM**
Batch	✓			✓	✓	✓	✓	✓	✓
Online	✓	✓	✓	✓	✓	✓		✓	✓
Interactive application development	✓	✓	✓	✓	✓	✓	✓	✓	✓
Query/Report Writing	✓	✓	✓						

Notes:

1 For this column:

- **Base** refers to the database manager components that support batch applications, the preprocessors, and the utilities including the database services (DBS) utility.
- **ORA** (online resource adapter) refers to the DB2 Server for VSE support for transaction processing (CICS) environments.
- **ISQL** refers to the terminal user query and report-writing facilities.

Virtual Storage Requirements

The size of the partition running the database manager is your primary design consideration for determining virtual storage requirements. Virtual storage requirements of other components are less significant as they are smaller or transient in nature.

Several factors contribute to the virtual storage requirements of the partition. The dominant ones are the sizes of the buffer pools (used for directory and data), the number of concurrent users to be supported, the complexity of the SQL requests, and the additional storage required for the optional DRDA support. (The DRDA support makes DB2 Server for VSE data accessible to users equipped with the DRDA remote unit of work application requester function. For more information, see Chapter 14, “Using a DRDA Environment,” on page 313.)

Refer to the *DB2 Server for VSE Program Directory* for the recommended partition size, and for detailed formulas for calculating virtual storage requirements.

Hardware Requirements

Hardware requirements include real storage, DASD space, tape, and display terminals.

Real Storage Requirements

You need not allocate any real storage for the database manager over and above what is already defined for the partition. However, if more real storage is available, there is less paging, thus improving performance.

The VSE guest sharing facility requires 40 kilobytes of real storage for each database communication link.

DASD Space Requirements

DASD space requirements are discussed under the following categories: libraries, database data sets, and starter database. If you use the accounting facility, you can direct its output to DASD or tape. For guidelines on estimating DASD space requirements for the accounting facility, see Chapter 10, “Using the Accounting Facility,” on page 187.

Libraries: Refer to the *DB2 Server for VSE Program Directory* for the requirements of the DB2 Server for VSE library on various devices.

Database Data Sets: A database requires a minimum of three VSAM datasets:

- A directory extent, to hold internal control information for the database.
- Either one, two, or four log extents, to hold recovery information. Only one (known as the active log) is required, but defining an alternate log volume may prevent inappropriate log archives from occurring. Whether alternate logging is used or not, the use of dual log volumes on separate volumes is recommended, to protect against I/O errors on access to the log information.
- Database extents (dbextents), to hold the user data of the database. It is possible to have only one dbextent, but a typical database has several.

The directory and log extents are described in Chapter 2, “Planning for Database Generation,” on page 13, and database extents are described in Chapter 7, “Managing Database Storage,” on page 123.

The Starter Database: The A-type member ARISDBG, which comes with this product, contains IBM-supplied specifications for generating a starter database. This database consists of one directory data set, one log data set, and one user-data dbextent. You can later add more dbextents, up to a logical maximum size of about 4.6 gigabytes, using the information in “Adding Dbextents to a Storage Pool” on page 133.

It is recommended that you generate the starter database at the time of installation, and experiment with it in order to familiarize yourself with the database manager. You may then keep it as your production database. However, as your needs grow, you may find it necessary to transfer its contents to another database, which can be a major undertaking. Thus, once you are familiar with how it works, it is best to discard it and generate your own database by following the guidelines in Chapter 2, “Planning for Database Generation”.

The initial physical size of the starter database is predefined and will be about the same on all IBM storage devices. Figure 6 on page 10 shows the approximate cylinder allocations (or block allocations, in the case of FB-512) on various devices.

Starter Database Data Set Allocations	3375 Cyls	3380 Cyls	3390 Cyls	9345 Cyls	FB-512 Blocks
Directory	53	34	29	38	31,620
Log	13	8	7	9	9,600
Data Extent					
Minimum	38	24	21	27	28,800
Recommended	121	77	65	85	92,400
Total Allocations					
Minimum	**104**	**66**	**57**	**74**	**70,020**
Recommended	**187**	**119**	**101**	**132**	**133,620**

Figure 6. Recommended Starter Database DASD Sizes

This starter database must be able to fit in a single dbextent. If you do not have enough DASD, you will not be able to use the IBM-supplied specifications, and will have to generate your own database at the time of installation. If you want to define the equivalent of the starter database on the devices shown in Figure 6, you must define multiple dbextents on multiple volumes.

If you are migrating from a previous release of the database manager, you already have at least one database, so generating the starter database is optional. The advantage of doing so is that you can use it as a test database to verify your installation, but the disadvantages are the work involved and the necessary DASD allocations. Thus to deal with migration needs, the database manager provides allocations for generating a starter database that is large enough to hold the initial database components (for example, HELP text, catalog tables, and FORTRAN packages), but not much else. Figure 6 also shows the data set sizes for a minimum starter database.

VSAM Catalogs: The database manager must have a VSAM master catalog and optionally, a VSAM user catalog. Each of the database data sets (the directory, the logs, and the dbextents) must be cataloged in either a user catalog or the master catalog.

Tape Requirements

One tape drive is required for installation. Once the database manager is running, tape drives are only required for the following activities:

- Database archive and log archive processing (both creating the archive and restoring the database from the archive) to support recovery from DASD failures
- Unloading and reloading data into the database using the DBS utility
- Holding the output of the trace facility
- Holding the output of the accounting facility

For all of these facilities except archiving, you can use DASD instead of tape.

Also, with the exception of accounting output, the database manager does not require the continuous use of any tape drive: tape mounts are requested when needed. If you are using tape drives, you should have at least two to cover all your needs.

The database manager supports all tape drives that are supported by the operating system.

Display Terminal Requirements

A variety of display terminals are supported, including the larger screen sizes offered by some models of the 3278 and 3279 (or equivalent) devices. Since CICS is needed to provide terminal support for DB2 Server for VSE online applications, the terminal must be one that is supported by CICS.

You can direct ISQL-printed output to a terminal printer rather than the system printer. To use terminal printers with ISQL, update the CICS tables as described in the *DB2 Server for VSE Program Directory* manual.

Terminals and workstations that follow the line disciplines represented on VSE/Power remote job entry are also supported by ISQL.

Note: To display and print DBCS characters (for example, Japanese HELP text), a DBCS terminal and printer (for example, the IBM 5550 terminal) are required.

DBNAME Directory Requirements

The DBNAME directory is a required directory of all DBNAMEs accessible from the VSE system. It identifies:

- The system default application server and partition defaults
- All valid TPNs used to access the DB2 Server for VSE application server from remote application requesters
- Communications parameters required for a local VSE DRDA requester to access a remote DRDA Server over SNA or TCP/IP.

The DBNAME directory is contained in A-type source member ARISDIRD. IBM-supplied defaults are provided, but these can be changed. For more information, see "Setting Up the DBNAME Directory" on page 23.

DB2 Key Processing

When running on VSE/ESA 2.5 or later, DB2 Server for VSE is key-enabled. For information on setting up the DB2 key, see the *DB2 Server for VSE Program Directory*.

Chapter 2. Planning for Database Generation

As described in "The Starter Database" on page 9, when you first install the database manager you should generate an initial database using the IBM-supplied specifications. This eases installation, and enables you to gain experience with the system.

However, once you know how to work with this database, you will probably want to discard it and create several databases that are tailored to your own needs. This chapter describes the parameters that are set at the time of database generation, and presents some general design considerations.

If you are migrating from an earlier version of the database manager, then instead of reading this chapter go to Chapter 3, "Planning for Database Migration," on page 39.

The database-generation process does not require definition of any data specifics; it merely establishes the potential capacity of the database. Some of the capacity-planning decisions require knowledge of the data and application requirements of your users. For example, to estimate how big the database will become, you need to know the potential number of tables that will be stored, and the storage requirements of those tables. To obtain this information, consult with the person responsible for the data and application requirements for the database. Also refer to the *DB2 Server for VSE & VM Database Administration* manual.

Setting Up the DB2 Product Key

When running on VSE/ESA 2.5 or later, DB2 Server for VSE is key-enabled. For information on setting up the DB2 key, see the *DB2 Server for VSE Program Directory*.

Database Generation Parameters

Planning for the generation of a database entails establishing logical and physical limits for its capacity, and setting its initial DASD allocations.

The parameters that you must establish at this time are summarized in Table 2 on page 14. This figure also shows the IBM-provided values used for the starter database.

Note: The parameters that have a `Yes` entry in the **Fixed** column must be established during generation of the database, and cannot be changed for the lifetime of the database. Also note that some parameters are established by running the VSAM IDCAMS program, whereas others are established by input to an IBM-supplied job called ARISQLDS.

Following the figure is a discussion of how to set these parameters, and of the issues to consider when setting them.

Table 2. Database Parameters Set at Database Generation Time

Parameter	Default	Minimum	Maximum	Starter Database	Fixed	Set by
Database directory size	None	2 tracks	1 volume	34 cylinders	No	IDCAMS
Log data set (or data sets) -Size (each) -Number	None None	1 cylinder 1	524,287 4Kb pages 4 volumes	8 cylinders 1	No	IDCAMS
Maximum number of storage pools (MAXPOOLS)	32	1	999	256	Yes	ARISQLDS
Maximum number of dbextents (MAXEXTNT)	64	1	999	256	Yes	ARISQLDS
Maximum number of dbspaces (MAXDBSPC)	1000	10	32000	10240	Yes	ARISQLDS
Catalog dbspace (PUBLIC.SYS0001) Size (4 kilobyte pages)	None	128	8388607	12800	Yes	ARISQLDS
First package dbspace (PUBLIC.SYS0002) Size (4 kilobyte pages)	None	128	8388607	2048	Yes	ARISQLDS
HELP text dbspace (PUBLIC.HELPTEXT) Size (4 kilobyte pages)	None	2304	8388607	8192	No	ARISQLDS
ISQL dbspace (PUBLIC.ISQL) Size (4 kilobyte pages)	None	128	8388607	1024	No	ARISQLDS
SAMPLE dbspace (PUBLIC.SAMPLE) Size (4 kilobyte pages)	None	512	8388607	512	No	ARISQLDS
Internal dbspaces -Size (each) (4 kilobyte pages) -Number	None None	128 5	8388607 31997	1024 80	No	ARISQLDS
Initial dbextents -Size (each) -Number	None None	1 cylinder 1	1 volume 999	77 cylinders 1	No	IDCAMS

Notes:

1. The cylinder specifications listed above for the starter database are for IBM 3380 storage devices. Make the appropriate adjustment for your storage devices.
2. PUBLIC means that the dbspace is publicly owned.

Defining Database Directory Size

The DB2 Server for VSE directory (called BDISK) contains control information and page tables for mapping dbspace page references to physical DASD locations. Its size determines the maximum number of dbextent pages and the number of page table entries that can be supported by the database being generated.

Use the VSAM DEFINE CLUSTER command to define the BDISK data set. The directory size is established by the TRK, CYL, or BLK parameter. If necessary, you can later expand the directory to hold more dbspace pages, or more dbspace and dbextent pages. Refer to “Expanding the Database Directory” on page 128 for more details.

Table 3 shows the recommended cylinder (or block) allocations for various DASD device types, based on assumed maximum database sizes.

Table 3. Recommended Directory Allocations for Various Database Sizes

Directory Space for Various IBM Storage Devices					
Maximum Database Size	3375	3380	3390	9345	FB-512 BLOCKS
10 megabytes	TRK(3)	TRK(3)	TRK(3)	TRK(4)	BLK(124)
50 megabytes	TRK(7)	TRK(6)	TRK(6)	TRK(7)	BLK(310)
100 megabytes	CYL(1)	TRK(11)	TRK(10)	TRK(12)	BLK(496)
500 megabytes	CYL(5)	CYL(4)	CYL(4)	CYL(6)	BLK(2232)
1 gigabyte	CYL(10)	CYL(8)	CYL(7)		BLK(4480)
2 gigabytes	CYL(19)	CYL(16)	CYL(14)	CYL(21)	BLK(8866)
4 gigabytes	CYL(38)	CYL(32)	CYL(27)	CYL(42)	BLK(17696)
5 gigabytes	CYL(47)	CYL(40)	CYL(34)	CYL(52)	BLK(22080)
10 gigabytes	CYL(92)	CYL(80)	CYL(68)	CYL(101)	BLK(44144)
50 gigabytes	CYL(459)	CYL(400)	CYL(337)	CYL(504)	BLK(220286)

Note: The values in this table apply when the defaults are used for MAXPOOLs, MAXDBSPC, and MAXEXTNT. These parameters are described in “Establishing Database Capacity Parameters” on page 17.

Use Table 3 to choose the initial directory size. Detailed information for generating its values is contained in Appendix B, “Estimating Database Storage,” on page 341. When estimating the maximum database size, include the sizes of the public, private, and internal dbspaces.

The directory data set for the starter database supports about 4.9 gigabytes of data. This includes space for internal dbspace definitions so the actual space supported for public and private dbspaces is about 4.6 gigabytes.

Directory Allocation Considerations

Maximum Database Size: The directory data set cannot extend beyond a single volume; therefore, the maximum database size is limited by the single volume capacity of the device type used. The absolute maximum size for a database is either 64 gigabytes or the limit imposed by the device type, whichever is smaller. For the limits imposed by various devices, see Table 38 on page 341 and Table 39 on page 342.

Placement of Directory: The directory data set will be used extensively by the database manager for resolution of data addresses. Thus, you should not allocate it to a volume that will contain either the log data sets or heavily used data dbextents. Instead, place it on a separate volume to avoid device contention.

If DASD is limited on your system and the directory must share a volume with data dbextents, put it on a volume with a dbextent that contains infrequently referenced data. For example, sharing a volume with private dbspaces or historical data is preferable to sharing one with public dbspaces or current, highly active data.

Defining the Database Log

The database manager requires at least one log data set and can support four. It is recommended that you use two log data sets, one being the active log and the other one being an exact copy of it.

The log data sets contain information, recorded during database processing, that is used to support database recovery facilities. This includes control information (for example, COMMIT statement and checkpoint records) and the specifics of database changes (for example, inserts, updates, and deletes).

If you define more than one log data sets they must be exactly the same size. Do not define them on different device types because it is almost impossible (because of rounding) to get identically sized data sets using space allocation algorithms.

To establish the size of the log data sets use the VSAM DEFINE CLUSTER commands for LOGDSK1 and (optionally) LOGDSK2, ALTLGD1, and ALTLGD2. The size you specify will depend on the use of the database and on the type of recovery capabilities you want. If you underestimate this size at database generation time, you can redefine it afterwards, as described in "Log Reconfiguration" on page 171.

Log Size Considerations

The log size depends on the number of changes that you expect will be made to the database and on whether or not you plan to use archiving facilities. If either database or log archiving is enabled, the log must be large enough to hold all the logging done between archives; otherwise it need only be large enough to hold the logging done in a few hours.

Note: If you are putting dbspaces in nonrecoverable storage pools, keep in mind that only minimal logging is done for them, so the following log size considerations would not apply to those dbspaces.

Log Size without Archiving: If you run the database manager without the archiving facilities (LOGMODE=Y or N), log space is reclaimed as applications finish and checkpoints of the database are taken. Usually, this occurs every few seconds or every few minutes. Many uses of the database manager can be supported by a log size of only one or two cylinders; however, a long-running application may require more log space.

Typically, the largest demand for log space is online loading or reorganization jobs. These jobs run longer than most applications and cause a lot of logging to occur.

A starting estimate for the initial log size is twice the space requirements of your largest dbspace. If you have one exceptionally large dbspace, you can disregard it and use the size of the next largest dbspace. The data in the largest dbspace can be loaded and reorganized offline with logging inhibited.

Log Size with Archiving: If you are using the archiving facilities (LOGMODE=A or L), log space is not reclaimed until an archive is taken. That is, log space is not reused between archives of the log or database. Typically, you would only archive the database once or twice a week. You may choose to do log archiving more frequently, depending on database usage.

To estimate the size of the log, consider the amount of logging that will occur between archives. A useful approach is to estimate the percentage of data that will be generated, deleted, and changed over one archive period as follows:

```
logsize estimate =  (percentage generated
                     + percentage deleted
                     + percentage changed x 2)
                    x database size
```

For example, assume that in a one-week period the database size grows by 5% but also shrinks by 4%, and that 6% of the database (rows) are changed. Your estimate for the log size would be:

```
logsize estimate =  .21 x database size
```

If your database size were 100 megabytes and you wanted an archive period of one week, your log size estimate would be:

```
logsize estimate =  21 megabytes
```

This is approximately 30 cylinders of an IBM 3390 DASD device.

Logging Generated by Loading: The log requirements for processing the DBS utility DATALOAD and RELOAD commands in multiple user mode are:

- If the NEW option is used: enough space to hold the log entries for all table rows to be inserted
- If the PURGE option is used: enough space to hold the log entries for all table rows to be deleted as well as for all rows to be inserted.

The log space consumption caused by these operations can be avoided by running the DBS utility in single user mode with LOGMODE=N specified, or by using the COMMITCOUNT option to force periodic checkpoints in multiple user mode.

Placement of Logs: Like the directory data set the log data sets are frequently referenced during processing. To avoid device contention, they should reside on separate volumes from the directory or heavily used dbextents.

Placement of Dual Logs: If dual logging is defined, place the logs on separate volumes. If they were allocated to the same one, loss of that volume would cause the loss of both logs, thus defeating the purpose of dual logging.

Establishing Database Capacity Parameters

The MAXPOOLS, MAXEXTNT, MAXDBSPC, and CUREXTNT keyword control statements can be specified on control card input to database generation (done by program ARISQLDS with the STARTUP=C initialization parameter). The first three of these statements are optional. The last one must be specified.

The MAXPOOLS, MAXEXTNT, and MAXDBSPC values are fixed when the database is generated: once defined, they cannot be changed for its lifetime. To avoid future limitation problems, it is recommended that you set them to the allowed maximums. This will take about 1 cylinder of DASD on a 3380 device for the directory, and 280K virtual storage when the database manager is running.

Estimating MAXPOOLS

The MAXPOOLS specification determines the maximum number of storage pools that can be defined in the database. Storage pools control the location of data on DASD volumes - that is, what dbspaces are located on what volumes. You can make a generous estimate for MAXPOOLS, since the value specified results in only a small directory space allocation for each potential storage pool. You should plan on having one storage pool for each user group (or billing account), and one for each major application you expect the database to support.

Estimating MAXEXTNT

The MAXEXTNT controls the maximum number of dbextents that are defined to support the database being generated. Dbextents determine the physical allocation of DASD space for a storage pool.

Because a dbextent is a VSAM data set, it cannot span DASD volumes. This means that you need at least as many dbextents as volumes. You can, of course, define multiple dbextents on one volume. It also means that if you have a dbspace that spans multiple volumes, the corresponding storage pool requires multiple dbextents.

Because you should plan to support multiple dbextents for each storage pool and you should be prepared to extend most, if not all, of your planned storage pools, MAXEXTNT should be much larger than MAXPOOLS. Your estimate for it can be generous because this value results in only a small directory space allocation for each potential dbextent.

Estimating MAXDBSPC

MAXDBSPC controls the maximum number of dbspaces, including internal dbspaces, that can be defined for the database. See “Determining the Internal Dbspace Requirements” on page 20. A dbspace is a logical allocation of database space for holding one or more tables and their indexes. A dbspace is assigned to a storage pool when it is defined and draws on the actual DASD space available in that storage pool on an as-needed basis. Typically, dbspaces are defined to support private space allocations for individual users and space allocations for specific applications; thus, the number of dbspaces required generally depends on the number of users and the number of tables needed for applications. Each user probably requires from one to five private dbspaces over the lifetime of the database, and each application requires, at most, one dbspace for each table being accessed. For performance reasons, one table per dbspace is recommended.

As with the previous two parameters, your estimate for MAXDBSPC can be generous, because the value you specify will result in only a small allocation of directory space for each potential dbspace.

Estimating CUREXTNT

CUREXTNT determines the number of dbextents defined during database generation. This number should be sufficient to support your initial storage requirements. You can add more dbextents after database generation.

Establishing Initial Dbspace Requirements

Determining the System Dbspace Requirements

Any public dbspace that has SYS as the first three characters in its name is reserved for system use only. The system dbspaces established at database generation time are PUBLIC.SYS0001, PUBLIC.SYS0002, PUBLIC.HELPTEXT, PUBLIC.ISQL, and PUBLIC.SAMPLE.

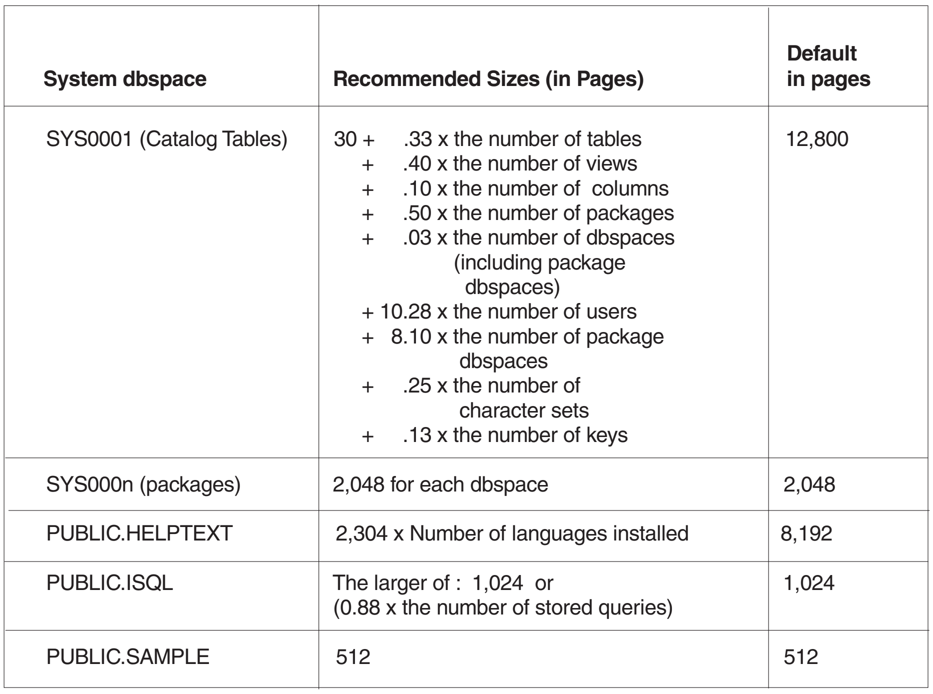
This section presents only general concepts related to setting the initial dbspace sizes. For more information, see “Specifying Initial Dbspaces” on page 223 and Appendix B, “Estimating Database Storage,” on page 341.

- PUBLIC.SYS0001 holds the database catalog tables. The size required for it varies considerably, depending on factors such as the number of tables, columns, indexes, views, and users in the database. For guidelines, see “Estimating SYS0001 Dbspace Requirements” on page 344.

  Note: Physical space is not actually consumed until required, so you can afford to define the SYS0001 dbspace to be very large. Be generous: this dbspace cannot be dropped or recreated after the database is generated. If you make it too small and SYS0001 runs out of usable space, you will have to regenerate the database which can be a considerable task.
- PUBLIC.SYS0002 holds the definitions of views and packages. This dbspace, which cannot be dropped or recreated after generation, can hold a combination of 255 views and packages. If you anticipate more views and packages than this, you can acquire additional dbspaces after database generation, as described in “Acquiring Dbspaces for Packages” on page 129.
- PUBLIC.HELPTEXT holds the online HELP tables. You will need 2304 pages for each IBM-supplied HELP text that you install. The starter database uses 8192 pages.
- PUBLIC.ISQL holds several tables; EXAMPLE.ROUTINE, SQLDBA.ROUTINE, and SQLDBA.STORED QUERIES. An allocation of 1024 pages should be enough for most uses. If you have many users or expect to make extensive use of the ISQL stored queries facility, consider increasing this. See “Estimating ISQL Dbspace Requirements” on page 353.
- PUBLIC.SAMPLE contains copies of the sample tables for ISQL users, to help them gain experience with using the database manager. Usually, every ISQL user has a copy of the sample tables. An allocation of 512 pages should be enough for all your users, but you can increase the size if you have many ISQL users. Alternatively, you can ask experienced ISQL users to drop their copies after they no longer need them to free space for new users’ tables.

The ARISDBU A-type member contains SQL statements to acquire the public dbspaces HELPTEXT, ISQL, and SAMPLE. If you want to increase their size, update the appropriate `ACQUIRE DBSPACE` statement in ARISDBU.

Except for PUBLIC.SAMPLE, the sizes that you establish for system dbspaces at database generation time can limit the logical capacity of your database. Because physical space is not actually used until required, you should establish large sizes for them. The large recommended sizes shown in Figure 7 on page 20 will support most uses of the database manager.



System dbspace	Recommended Sizes (in Pages)	Default in pages
SYS0001 (Catalog Tables)	30 + .33 x the number of tables + .40 x the number of views + .10 x the number of columns + .50 x the number of packages + .03 x the number of dbspaces (including package dbspaces) + 10.28 x the number of users + 8.10 x the number of package dbspaces + .25 x the number of character sets + .13 x the number of keys	12,800
SYS000n (packages)	2,048 for each dbspace	2,048
PUBLIC.HELPTEXT	2,304 x Number of languages installed	8,192
PUBLIC.ISQL	The larger of : 1,024 or (0.88 x the number of stored queries)	1,024
PUBLIC.SAMPLE	512	512

Figure 7. Guidelines for the Sizes of the System Dbspaces

Determining the Initial User Dbspace Requirements

When you generate the database, you need only consider the dbspace requirements for its initial use. To determine the initial user dbspace requirements, either consult with the database administrator or refer to the *DB2 Server for VSE & VM Database Administration* manual. The ADD DBSPACE facility can be used to add more later, up to the MAXDBSPC value.

For more information, refer to Chapter 7, “Managing Database Storage,” on page 123.

Determining the Internal Dbspace Requirements

The database manager uses internal dbspaces to process commands that require sort operations and to process views that require materialization. For information on sorting and materialization, see the *DB2 Server for VSE & VM Database Administration* manual.

The internal dbspaces are held until a COMMIT or ROLLBACK statement is issued; therefore, a single application may hold a number of internal dbspaces at one time. For example, if each SELECT needs an average of two internal dbspaces, and a certain program issues five SELECTs before issuing a COMMIT statement, then that program will hold 10 internal dbspaces. Internal dbspaces that are not in use take up minimal space (approximately 4 bytes of directory space for each page).

Allocate at least 30 internal dbspaces; more if your installation has interactive users. The exact number required depends on the number of logical units of work (LUWs) that are concurrently active and the amount of sorting and view materialization required in those LUWs. Because the number of NCUSERS is comparable to the number of concurrently active LUWs, as a guideline, in addition to the minimum of 30, you may want to provide 10 internal dbspaces for each NCUSER (see the description of the NCUSERS parameter on “NCUSERS” on page 55). After the database has been generated, you can always add more internal dbspaces by using the ADD DBSPACE function. All internal dbspaces (and their storage pool assignments) are redefined on each run of this function.

The physical placement of the internal dbspaces affects performance, especially when you perform a sort operation on a large table. You should place internal dbspaces in their own storage pool, and use multiple dbextents over multiple devices. There are several ways of doing this. Suppose you had 300 3380-type cylinders for internal dbspace dbextents, you could use one of these strategies:

1. Make the first dbextent small (less than 100 cylinders), and each succeeding dbextent twice the size of the preceding one. For example, have dbextents that are 20, 40, 80, and 160 cylinders in size.
2. Graduate the sizes of the dbextents. For example, have dbextents that are 10, 20, 30, 40, 50, 60, and 90 cylinders in size. The last dbextent is extra large so that unusually large sorts can be accommodated.
3. Have several small dbextents and a few big ones. For example, have five dbextents of 20 cylinders each, and two of 100 cylinders.

The purpose of all these strategies is to spread input/output activity over more devices as the size of a sort increases. The strategy you adopt determines how many dbextents a sort requires. With the first strategy, a sort requiring 60 cylinders uses two dbextents. With the second and third strategies, the same sort requires three dbextents. Use a strategy that is suitable for your organization.

Sorting is done for ORDER BY, GROUP BY, join, CREATE INDEX, or UNION operations. The internal dbspaces must be large enough to hold the rows being sorted. For example, if an ORDER BY operation is requested using all the columns of an entire table, the internal dbspace must be large enough to hold the whole table. Less space is required if all the columns are not selected. During index creation, space is required only for the key columns. To calculate the required size of an internal dbspace, use the formula (KEYSIZE + 8 bytes) * ROWCOUNT. Make the internal dbspaces large enough to hold the largest table or query result you want to be able to sort. The dbspace size estimates are discussed under Appendix B, “Estimating Database Storage,” on page 341.

The number of internal dbspaces required also depends on the planned usage of the system. Fewer are needed for preplanned application processing than for dynamic query processing, as query users usually hold dbspaces longer than do preplanned applications.

Internal dbspaces can also be stored on a virtual disk. Only use virtual disks for internal dbspaces because information on a virtual disk is lost when the database is restarted. For more information on virtual disk support, see the *DB2 Server for VSE & VM Performance Tuning Handbook* manual.

Determining Initial Dbextent Requirements

Sufficient space must be allocated during database generation to support your initial dbspace data storage requirements. You must define at least one dbextent for each storage pool that initially contains dbspaces. The specific amount to allocate for each storage pool depends on the following considerations:

- System dbspace support

System dbspaces are heavily used, so they should not share their storage pool (storage pool 1) with heavily used user dbspaces. Until you gain experience with your data, do not put user dbspaces in the same storage pool as system dbspaces.

You should undercommit storage pool support for the SYS0001 and SYS0002 dbspaces. If the catalog tables grow significantly, you can later allocate an additional dbextent, probably on a separate device, to avoid excessive device contention on catalog access.

Storage pool support for PUBLIC.HELPTEXT should be large enough to hold the HELP tables; PUBLIC.ISQL must be large enough to hold your initial needs for stored queries; and PUBLIC.SAMPLE should be large enough to hold the number of sample data tables needed.

- End user dbspace support

  Dbspaces for use primarily by end users should be supported by one or more storage pools. Public and private dbspaces can share a storage pool; however, you may want to manage space allocation differently for these two cases.

  A recommended approach to storage pool support for end user data is to define more dbextent space than is needed to support your initial dbspace definitions. This approach is called overcommitting, and ensures that end user space requirements can be accommodated as existing users need more space or more users are added to the system.

  If your installation plans to bill users for DASD storage space, you may want to consider separate storage pools for different user groups (or account numbers).

  Note: You can also use statistics from the SYSTEM.SYSDBSPACES catalog table to achieve this.

- Dbspace support for applications

  Storage pool support of dbspaces for use primarily by application programs varies, depending on the nature of the data and the storage management technique. In general, consider using different storage pools for different applications, and undercommitting storage pool support for application dbspaces.

  The dbspaces for applications should be defined to be larger than is believed necessary, to avoid later reorganization because of data growth. If you do this, storage pool requirements are smaller than the dbspace sizes indicate. The initial storage pool allocations should be large enough to cover initial loading of the data plus growth over the next planning period (for example, six months or a year).

- Internal dbspace support

  Storage pool support for internal dbspaces should be undercommitted, since you probably do not need storage to support all internal dbspaces at the maximum size. As a rough estimate, the storage pool for internal dbspaces should have enough DASD space available to hold data for three internal dbspaces (at the internal dbspace size specified at database generation).

  Storage space for internal dbspaces is taken from the storage pool assigned at database generation time. In general, this storage pool should not be used for system dbspaces or other heavily used dbspaces. Consider using a separate storage pool just for internal dbspaces.

For more information on storage organization techniques, see Chapter 7, “Managing Database Storage,” on page 123.

Choosing an Application Server Name

In planning for database generation, you choose two names for your database.
- The first name is the *mapped DBNAME* (DBNAME).
- The second name is the *basic DBNAME*. This is the VSE subsystem application identifier, APPLID. It identifies the application server subsystem to VSE.

These names are specified in the DBNAME directory. In this directory, the mapped DBNAME is mapped to the basic DBNAME. Details regarding this directory can be found in “Setting Up the DBNAME Directory.”

When the application server is started, you must supply the mapped DBNAME. This value is now the *server name*, and is stored in the CURRENT SERVER register.

Attention: The mapped DBNAME is the name the application requesters should specify when connecting to the application server.

The mapped DBNAME must be from one to 18 characters. It should start with an uppercase alphabetic character. The remaining characters can be alphabetic, numeric or underscore characters. The mapped DBNAME must not be prefixed with “SYSARI”. It should be unique within a set of networks that are interconnected, and be defined and stored in the DBNAME directory in the production library.

The basic DBNAME has a length of eight alphanumeric characters, and must be unique within the VSE system because it is used to identify the application server subsystem to the VSE system. The basic DBNAME is either the VSE APPLID (SYSARI0x) or, in the case of guest sharing, the VM resource identifier (RESID). There are 36 reserved basic DBNAMEs (SYSARI00 to SYSARI09, SYSARI0A to SYSARI0Z), which must be reserved for VSE application servers only. If the basic DBNAME is not prefixed with “SYSARI”, it is assumed to be a VM application server and must be defined with a `SET APPCVM TARGET` command if it is on a remote system. In this case, the basic DBNAME defined in the DBNAME directory must be identical to the VM RESID.

You must also decide whether the application server will be accessed from remote application requesters. If the DRDA environment will be used, you must also choose a CICS transaction program name (TPN) to represent the application server.

Note: When using remote access, it is recommended that the system administrator ensure that server names are unique within a set of interconnected SNA networks. APPLIDs are predetermined. For more information on these requirements, see Chapter 14, “Using a DRDA Environment,” on page 313.

Setting Up the DBNAME Directory

The DBNAME directory is a **required** user-definable directory of **ALL** Local, Remote and Host VM application server names. A Local server executes in a partition in the VSE system. A Remote server exists external to the VSE system (and must be connected via SNA or TCP/IP). A Host VM server exists on the VM system on which the VSE system runs as a guest and DB2 Guest Sharing is used between the VSE requesters and the VM server.

In addition, the DBNAME Directory contains all Transaction Program Names (TPNs) used by remote application requesters to identify the local DB2 Server for VSE application servers which they want to access.

The IBM supplied default DBNAME Directory is contained in the A-type source member called ARISDIRD. It contains the default *mapped* DBNAME ″SQLDS″, which defines a Local server which is also the System Default DBNAME. It also contains the default Registered TPN (X'07F6C4C2), which points to the default DBNAME ″SQLDS″.

The DBNAME Directory consists of a number of *entries* that define all mapped DBNAMEs and all TPNs. Each mapped DBNAME can have an *ALIAS* name, to allow multiple entries (with different options) to specify the same mapped DBNAME. The Alias name defaults to the mapped DBNAME, if not specified. Each Alias name **MUST** be unique within the DBNAME Directory source file and this will be enforced.

There are four types of entries, as follows:

- **LOCAL:** This type of entry defines a server that physically exists on the same VSE system as the DBNAME Directory.
- **HOSTVM:** This type of entry is for Guest Sharing only and defines a server that exists on the Host VM system on which the VSE system is a guest.
- **REMOTE:** This type of entry defines a DRDA-conforming server on a system that is physically remote from the VSE system and is connected to the VSE system via an SNA or a TCP/IP communications network.
- **LOCALAXE:** This type of entry identifies a local mapped DBNAME and the TPN of a remote requester that can access this local DBNAME via the *APPC-to-XPCC Exchange* (AXE) CICS Transaction. This type of entry cannot have an Alias name.

The first three types of entries define mapped DBNAMEs, while the LOCALAXE entry defines the names of CICS ″AXE″ transactions and their target local servers.

The DBNAME Directory does not have a maximum size. It is searched sequentially and the first entry that matches the search argument is used. Where there are multiple default partition entries, the last default partition entry will be used. Therefore, it is possible to have overriding entries, but excessive overrides impact search performance and should be avoided.

The following four figures show the syntax of the four directory entry types. The keywords and values are defined after these figures.

Each entry consists of a number of lines, each of which specifies a single ′KEYWORD=value′. The first line MUST be the TYPE of the entry and the second line MUST be the DBNAME of the entry. Blank lines are ignored. If the first 2 non-blank characters of the line are ′*′, ′/*′, or ′--′ then the line will be ignored and can be used for comments.

```
►►──TYPE=LOCAL──DBNAME=local_database_name──APPLID=─SYSARI0──x──────────────────►

►─┬──────────────────────────┬─┬─────────────────────────────┬──────────────────►
  │             ┌─DBNAME─────┐ │ │                ┌─0──────────┐ │
  └─ALIAS──=────┴─alias_name─┴─┘ └─TCPPORT──=─────┴─portnumber─┴─┘

►─┬──────────────────┬─┬────────────────┬─────────────────────────────────────►◄
  │             ┌─N─┐ │ └─PARTDEF──=──pd─┘
  └─SYSDEF──=───┴─Y─┴─┘
```

Figure 8. LOCAL Type Entry

A Sample LOCAL Directory entry with only required lines:

```
TYPE=LOCAL
DBNAME=VSEDATABASE
APPLID=SYSARI00
```

```
►►──TYPE=HOSTVM──DBNAME=host_vm_database_name──RESID=vm_database_iucv_resid─────►

►─┬──────────────────────────┬─┬───────────────────────┬─┬────────────────┬────►◄
  │             ┌─DBNAME─────┐ │ │                ┌─N─┐  │ └─PARTDEF──=──pd─┘
  └─ALIAS──=────┴─alias_name─┴─┘ └─SYSDEF──=──────┴─Y─┴──┘
```

Figure 9. HOSTVM Type Entry

A Sample HOSTVM Directory entry with only required lines:

```
TYPE=HOSTVM
DBNAME=VMDATABASE3
RESID=VMRESID3
```

```
►►──TYPE=REMOTE──DBNAME=remote_database_name──┬───────────────────────────┬──┬──────────────────┬──►
                                              │           ┌─DBNAME─────┐  │  │            ┌─N─┐ │
                                              └─ALIAS──=──┴─alias_name─┴──┘  └─SYSDEF──=──┴─Y─┴─┘

►──┬──────────────────┬────────────────────────────────────────────────────────────────────────────►
   └─PARTDEF──=──pd───┘

►──┬──────────────────────────────────────────────────────────────────────────────────────────┬──►
   │      (1)                                                                                   │
   ├─SYSID─────────=──cics_appc_connection_name────REMTPN──=──remote_transaction_program_name──┤
   │        (1)                                                                                 │
   └─TCPPORT───────=──portnumber───┬─IPADDR──=──dotted_decimal_address──────┬──────────────────┘
                                   └─TCPHOST──=──remote_server_tcpip_host_name─┘

►──┬────────────────────────────────┬──────────────────────────────────────────────────────────►◄
   │             ┌─N─┐              │
   ├─PWDENC──=───┴─Y─┴──────────────┤
   │ ┌──────────────────────────────┤
   │ │              ┌─Y─┐           │
   │ ├─CONNPOOL──=──┴─N─┴───────────┤
   │ │ ┌────────────────────────────┤
   │ │ │             ┌─Y─┐          │
   │ │ ├─PWUPPER──=──┴─N─┴──────────┤
   │ │ │ ┌────────────────────────┐
   │ │ │ │              ┌─Y─┐     │
   │ │ │ └─CONNPOOL──=──┴─N─┴─────┘
```

Notes:

1 Either 'SYSID=' or 'TCPPORT=' must be specified, and both can be specified.

Figure 10. REMOTE Type Entry

A Sample REMOTE Directory entry with only required lines:

```
TYPE=REMOTE
DBNAME=REMOTEDB
SYSID=REMS
REMTPN=REMTRANSPROGRAMNAME
```

or

```
TYPE=REMOTE
DBNAME=REMOTEDB
TCPPORT=4865
IPADDR=98.76.54.32
CONNPOOL=Y
```

```
►►──TYPE=LOCALAXE──DBNAME=local_database_name──APPLID=─SYSARI0──x──────────────►

►─TPN=cics_axe_transaction_name──┬──────────────────────┬─────────────────────►◄
                                 │           ┌─N─┐      │
                                 └─PRIV──=───┴─Y─┴──────┘
```

Figure 11. LOCALAXE Type Entry

A Sample LOCALAXE Directory entry with only required lines:

```
TYPE=LOCALAXE
DBNAME=LOCALDB
APPLID=SYSARI0Q
TPN=WXYZ
```

These keywords and values are described below:

TYPE=
This value is required, it is the type of entry that follows and must be one of **LOCAL**, **HOSTVM**, **REMOTE** or **LOCALAXE**. It must be the first line of an entry.

DBNAME
This 1 to 18 character *mapped* name is required. It does not need to be unique within the file but it cannot begin with SYSARI. The first character must be alphabetic or ′#′, ″$″ or ″@″; other characters can be alphabetic, numeric, ″#″, ″$″, ″@″ or ″_″. It must be the second line of an entry.

ALIAS=′*alias*′
This identifies an alias database name for the mapped DBNAME; it is optional but if specified it **MUST** be unique within the file. An alias name is an 18 byte name and defaults to the mapped DBNAME. When the Directory is searched via a DBNAME, it is the *alias* name that is searched, but it is the *mapped* DBNAME that is used to connect to the server. Note that a LOCALAXE entry cannot have an alias name, because it is always searched via the TPN.

APPLID
This is the basic DBNAME and is the XPCC ″Application Name″ that corresponds to the DBNAME. It is required for LOCAL and LOCALAXE entries. There **must** be a one to one correspondence between each APPLID and DBNAME in the file. The value must be specified as `SYSARI0`**`x`**, where ″**x**″ is upper case A through Z, or 0 through 9.

TPN This is the CICS AXE Transaction Program Name that is used by remote requesters to access this local DBNAME server and is required for LOCALAXE entries only. This is a 4 character value consisting of upper case A-Z, 0-9, or ″#″, ″$″, ″@″, or ″_″. Alternatively, it can be specified as an 8 character hexadecimal value consisting of 0-9 or A-F.

PRIV This is a flag to determine if this TPN is a privileged user of a local server’s Real Agent. It must be ″Y″ or ″N″ and is optional only for LOCALAXE entries. Normally a Real Agent is released when a requester reaches the end of an LUW. A privileged TPN will retain exclusive use of the Real Agent until the end of the SNA or TCP/IP communications session. Note that privileged TPNs can cause contention problems for other users of the server.

PARTDEF
This option indicates that this DBNAME is the Partition Default for the partition(s) specified in the value. It is used when either a server or requester is started without specifying a DBNAME and is optional only for LOCAL, HOSTVM or REMOTE entries. A server started in the specified partition will use this entry’s DBNAME. A requester executing in the specified partition will connect to this entry’s DBNAME. It is a two character partition name, for example: ″BG″, ″F3″, ″H2″ or ″X*″. For dynamic partitions, the second character may be an asterisk (*) to indicate that this DBNAME is the default for any dynamic partition whose name begins with the first character. The default is blank, which means that this DBNAME is not the default for any partition.

SYSDEF
This option indicates that this DBNAME is the System Default. It is used when either a server or requester is started without specifying a DBNAME and no Partition Default is found; it is optional only for LOCAL, HOSTVM

or REMOTE entries. You must specify ″Y″ or ″N″, ″N″ being the default. Only one entry in the directory can be specified as the System Default and this restriction is enforced.

SYSID
This option specifies the CICS APPC Connection Name (of an entry in the CICS Terminal Control Table, defined by the CICS CEDA DEF CONNECTIONS command) that identifies the SNA connection with the remote system where the DRDA-capable DBNAME server resides. This is a 4 character value consisting of upper case alphabetic, numeric or ″#″, ″@″, ″$″ or ″_″ characters. The default is blank, which indicates that there is no SNA access to this DBNAME server. If SYSID is specified in this entry, REMTPN must also be specified. A REMOTE entry must specify SYSID or TCPPORT, or both.

SYSID is optional only for REMOTE entries.

REMTPN
This is the 1-32 character Remote Transaction Program Name of the remote server which is optional only for REMOTE entries. This value is not checked and there is no default. It **must** be specified if the SYSID option is specified.

TCPPORT=′*port***′**
This option only applies to LOCAL or REMOTE entries.

For LOCAL entries, it identifies the TCP/IP Port Number to be used by the LOCAL server to accept incoming TCP/IP connections. This value can be overridden by the local server ′TCPPORT′ Start Up Parameter. The default is minus one, meaning that the TCP/IP support to be used will be determined by the Server. Valid values range from zero through 65,535, with zero indicating TCP/IP support is NOT to be used by this server.

For REMOTE entries, it identifies the TCP/IP Port Number to be used by the local requester when making a connection to the REMOTE server. Valid values range from minus one through 65,535. Minus one and zero both mean that no port number was specified in this DBNAME directory entry, which means that no TCP/IP communications is available to this remote server. If this parameter is specified for REMOTE entries, IPADDR or TCPHOST must also be specified. A REMOTE entry **must** specify SYSID or TCPPORT, or both.

IPADDR=′port′
This option only applies to REMOTE entries and identifies the TCP/IP Dotted-Decimal Address of the REMOTE server. The value must be specified as ″nnn.nnn.nnn.nnn″, where the periods (″.″) are required delimiters and each ″nnn″ value is a decimal number between 0 and 255. If this parameter is specified, **TCPPORT** must also be specified, and **TCPHOST** must NOT be specified.

TCPHOST=′*host_name***′**
This option only applies to REMOTE entries and identifies the TCP/IP Host Name of the REMOTE server. It is a 1-64 character Host Name and must be known to the TCP/IP network. This value is not validated and there is no default. If this parameter is specified, **TCPPORT** must also be specified, and **IPADDR** must NOT be specified.

PWDENC
This option only applies to REMOTE entries. You must specify ″Y″ or ″N″, ″N″ being the default. Specify a value of ″Y″ to encrypt the CONNECT

password. The target database server must support decryption of the password. A value of ″N″, or the absence of this option, will result in the password being sent to the server as plain text.

PWUPPER
The PWUPPER parameter is valid only for REMOTE entries for remote servers. You must specify ″Y″ or ″N″, ″Y″ being the default. Specify a value of ″Y″ to generate the passwords in uppercase. A setting of ″N″ will allow the password to be sent to the remote server as it is entered.

CONNPOOL
This option only applies to REMOTE with TCPPORT entries for online users. You must specify ″Y″ or ″N″, ″Y″ being the default. A value of ″Y″ or the absence of this option, activates the CONNECTION POOLING feature upon CIRB/CIRA entry for the target database. Specifying a value of ″N″, will result in deactivating this feature while accessing the target database. Recommended for online users whose applications are database switching intensive between remote databases connected over TCP/IP and other databases. Recommended also for online users who have applications with frequent SQL CONNECT statements across LUWs to the same remote database.

The IBM-Supplied DBNAME Directory

The following example shows the IBM-supplied default DBNAME Directory, including the System Default local server DBNAME of ″SQLDS″ and the default registered DRDA AXE TPN Name X'07F6C4C2, which maps to a DBNAME of ″SQLDS″ and an APPLID of ″SYSARI00″.

You must **not** delete either of these supplied entries and you should insert any additional entries preceding the two supplied entries. However, if you add an entry for a server that is to be the System Default entry (for example, the entry contains the SYSDEF=Y option), you **must** remove the SYSDEF=Y option from the supplied entry, as only one entry can use that option.

```
*
TYPE=LOCALAXE
DBNAME=SQLDS
APPLID=SYSARI00
TPN=07F6C4C2
*
TYPE=LOCAL
DBNAME=SQLDS
APPLID=SYSARI00
SYSDEF=Y
```

Updating the DBNAME Directory

The DBNAME Directory source file is an A-type member ARISDIRD in the production library. **All** local server, remote server and host VM server DBNAMEs **must** be identified in this member.

Place your new entries before the IBM-supplied entries. Remember to remove the SYSDEF=Y option from the IBM-supplied entry if you define a different System Default entry. Catalog your changed member back into the production library. If you catalog your changed member under a different name than ″ARISDIRD″, be sure to update the ″PARM=″ field of the EXEC statement in the ARISBDID JCL before executing the JCL.

The member must then be processed by the IBM-supplied ARISBDID Job Control Language member to catalog the DBNAME Directory Service Phase ("ARICDIRD.PHASE") into the production library. For more information on this process, see the *DB2 Server for VM Program Directory*. Any errors or warnings during this processing will appear in the SYSLST listing.

Sample DBNAME Directory

The following is a sample DBNAME Directory, with explanatory notes:

```
TYPE=LOCAL
DBNAME=SQLDB4_SANJOSE
APPLID=SYSARI03
PARTDEF=F4
```

If a server is started in partition F4 and no DBNAME start up parameter is specified, then this entry is used. Likewise, if an application is executing in partition F4 and issues an SQL CONNECT statement without a 'TO' clause, it will use this entry and access DBNAME 'SQLDB4_SANJOSE'.

```
TYPE=LOCAL
DBNAME=SQLDB1_NY
APPLID=SYSARI02
SYSDEF=Y
```

This entry is the System Default entry. Any application executing in a partition that is **NOT** identified in this directory will access DBNAME 'SQLDB1_NY'.

```
TYPE=LOCAL
DBNAME=SQLDB3_TOR
ALIAS= SQLDB3_TOR
APPLID=SYSARI0A
PARTDEF=BG
```

Applications executing in partition BG will access DBNAME SQLDB3_TOR.

```
TYPE=LOCAL
DBNAME=SQLDB3_TOR
ALIAS= SQLDB3_TORX
APPLID=SYSARI0A
PARTDEF=X*
```

Applications executing in a dynamic partition with a partition name beginning with 'X' will access DBNAME SQLDB3_TOR. Note the use of the ALIAS= keyword above. As the previous entry used the alias 'SQLDB3_TOR', and all alias names **MUST** be unique, this entry must use a different alias name, even though the DBNAMEs in both entries are equal.

```
TYPE=LOCALAXE
DBNAME=TORONTO_LAB
APPLID=SYSARI07
TPN=SQL1
PRIV=Y
```

When a remote DRDA requestor communicates with CICS and passes a TPN of 'SQL1', CICS starts transaction 'SQL1' (after validation). Transaction 'SQL1' will connect to this entry's APPLID via XPCC, which is DBNAME 'TORONTO_LAB'. Also, because the "PRIV=Y" option is specified, CICS transaction 'SQL1' has extended use of the server's Real Agent until the DRDA conversation ends.

```
TYPE=HOSTVM
DBNAME=VMDATABASE1
RESID=VMDB1
```

This entry identifies a DB2 Server for VM database server that executes on the VM system under which this local VSE system is running as a guest. Any local requester that connects to DBNAME ′VMDATABASE1′ will access the VM server via the Guest Sharing facility.

```
TYPE=REMOTE
DBNAME=SQLMACJR
ALIAS=TOKYO
SYSID=VMC3
REMTPN=JRSERVER
TCPPORT=27
IPADDR=94.83.72.161
```

This entry identifies a remote DRDA-capable server with a DBNAME of ′SQLMACJR′ and an alias of ′TOKYO′. It can be accessed by CICS requesters via SNA using a Remote Transaction Program Name (REMTPN) of ′JRSERVER′. It can also be accessed by CICS and Batch requesters via TCP/IP using IP Address ′94.83.72.161′ and Port number ′27′. Note that Batch requesters cannot access remote servers via SNA.

CICS CEDA DEF CONNECTIONS Command for a Remote Entry

Figure 12 shows an example of the CICS CEDA DEF CONNECTIONS command used to define the connection that matches the remote entry for ″DBNAME=SQLMACJR″ in the example above.

```
 Connection     : VMC3
 Group          : DRDA
CONNECTION IDENTIFIERS
 Netname        : OECGW001
 INDsys         :
REMOTE ATTRIBUTES
 REMOTESystem   :
 REMOTEName     :
CONNECTION PROPERTIES
 ACcessmethod   : Vtam              Vtam | IRc | INdirect
 Protocol       : Appc              Appc | Lu61
 SInglesess     : No                No | Yes
 Datastream     : User              User | 3270 | SCs | STrfield | Lms
 RECordformat   : U                 U | Vb
OPERATIONAL PROPERTIES
 AUtoconnect    : Yes               No | Yes | All
 INService      : Yes               Yes | No
SECURITY
 SECURITYNAME   :
 ATTACHSEC      :                   Local | Identify | Verify
 Bindpassword   :                   PASSWORD NOT SPECIFIED
```

Figure 12. Define Remote Connection

Choosing the Application Server Default CHARNAME and CCSID

The *application server default CHARNAME* is set using the CHARNAME initialization parameter. The database manager uses the CHARNAME value to determine the classification table and translation table which are used to identify valid characters and to determine how to fold lowercase characters to uppercase. For more information on the CHARNAME initialization parameter, see “CHARNAME” on page 51.

The CHARNAME parameter also specifies the *application server default coded character set identifier (CCSID)*. For a newly installed database, the application server default CHARNAME is INTERNATIONAL, and the application server default CCSID is 500. For a migrated database, the application server default CHARNAME is ENGLISH, and the application server default CCSID is 37. The application server default CCSID is the value of CCSIDMIXED if it is not zero, otherwise it is the value of CCSIDSBCS. Refer to "CCSID Conversion" on page 245 and "Determining CCSID Values" on page 248 for more information on CCSIDs.

If you use DBCS characters, you need to use a mixed CCSID as the application server default. A mixed CCSID has both an SBCS component CCSID, and a DBCS component CCSID. For more information, see Table 21 on page 247.

The application server default CCSID value is used for the following:
- The CCSID that SQL statements are converted to for processing by the relational data system (RDS) component
- The CCSID of constants (including hexadecimal constants) which are part of the SQL statement processed by the RDS component

Depending on the *application server default subtype* value (that is, the CHARSUB value), the application server default value for CCSIDMIXED or CCSIDSBCS is used for the following:
- The CCSID of special registers which represent character data (for example, CURRENT USER and CURRENT DATE)
- The CCSID of the results of the scalar functions CHAR, DIGITS, and HEX
- The CCSID of the character representation of datetime values (for DRDA protocol, this is always the CCSIDSBCS value)
- The CCSID of character columns created using the CREATE TABLE or ALTER TABLE statements (when the CCSID or subtype clause is not explicitly specified and when package defaults are not specified). See the *DB2 Server for VSE & VM Application Programming* manual for more details on package defaults.

It is important that you choose the correct default CHARNAME and CCSID for your installation. The goals of choosing the correct values are to ensure the integrity of character data representation, and to reduce the performance overhead associated with CCSID conversion. The application server and application requester should have the same CCSID value unless there is a specific reason for them to be different.

When the application server and application requester have different CCSID values, character conversion cannot be avoided. This conversion has an associated performance overhead. Performance degradation also occurs if the CCSID conversion causes a sargable predicate to become residual. For example, this can occur on a simple equals predicate like, T1.C1 = T2.C2. For this case, C2 was created prior to migrating to Version 3 Release 4 and has a CCSID of 37. C1 was created using Version 3 Release 4 with the application server default CHARNAME set to INTERNATIONAL (CCSID 500), As a result, since this predicate requires the CCSID conversion of the data in the columns, it is residual. For more information on performance, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

For example, if your application server is only accessed by local users whose terminal controllers are generated with code page 37 and character set 697 (CP/CS 37/697) for the US ENGLISH characters, then you should set the application server default CHARNAME to ENGLISH. This is because CP/CS 37/697 corresponds to the CCSID of 37 which corresponds to the CHARNAME of ENGLISH.

To eliminate unnecessary CCSID conversion, choose an application server default CCSID to be the same as the CCSID of the application requesters which access your application server most often.

The following is an example of how these two goals can be in conflict.

The situation has these characteristics:
- An application server is accessed by 5 application requesters which are local (that is, they have the protocol parameter set to SQLDS).
- This application server is also accessed by 100 application requesters which are remote (that is, they are using the DRDA protocol).
- The local application requesters have controllers which are defined with CP/CS 37/697 (this corresponds to CCSID 37).
- The remote application requesters use CCSID 285.

If the application server default CHARNAME is set to ENGLISH (CCSID 37), this keeps the data integrity for the local application requesters. However, CCSID conversion overhead is incurred for all remote application requesters who have CHARNAME UK-ENGLISH (CCSID 285).

If the application server default CHARNAME is set to UK-ENGLISH (CCSID 285), this will avoid the CCSID conversion overhead incurred for the remote application requesters, but will cause data integrity problems for the local application requesters. Certain characters will not be displayed correctly for local application requesters. For example, a British pound sign (£) will be displayed as a dollar sign ($).

These are the trade-offs to consider when choosing your application server default CHARNAME.

For more information on CCSIDs, see the *Character Data Representation Architecture Reference and Registry* manual.

Attention: Immediately following an installation, the application server CHARNAME is set to INTERNATIONAL and the CCSID is 500. Immediately following a migration, the application server CHARNAME is set to ENGLISH and the CCSID is 37. If you do not **choose** your own application server defaults, these settings may not be correct for your system.

For information on how to change the application server default CHARNAME and CCSID, see “Setting the Application Server Default CHARNAME and CCSIDs” on page 249. For a summary of the considerations for changing these values, see “Considerations when changing default CHARNAME and CCSID” on page 232.

Choosing the Application Server Default Character Subtype

The database manager supports three types of character data:
- SBCS
- Mixed
- Bit.

Note: Character refers to data types CHAR, VARCHAR and LONG VARCHAR in this discussion.

Each database has a default character subtype (that is, the CHARSUB value) which can be either SBCS (single-byte character set) or mixed (mixed single and double-byte character set). The default character subtype is the value used for the subtype attribute of any new character column that is created by either the CREATE TABLE statement or the ALTER TABLE statement. The default subtype is used if a subtype is not specified as a package default option or a preprocessing option, and is not specified explicitly using a subtype clause, or implicitly using a CCSID clause.

The CHARSUB value is also used for determining CCSIDs. For more information on CCSIDs, see "Choosing the Application Server Default CHARNAME and CCSID" on page 31, "CCSID Conversion" on page 245, and "Determining CCSID Values" on page 248. For information on how to change the default character subtype, see "Setting the Application Server Default Character Subtype" on page 253.

Choosing the Default CHARNAME and CCSID for Application Requesters

It is important that the appropriate *application requester default CHARNAME* and appropriate *application requester default CCSID* be chosen. The goals of choosing the correct values are to ensure the integrity of character data representation, and to reduce the performance overhead associated with CCSID conversion.

For example, if your terminal controller is generated with code page 37 and character set 697 (CP/CS 37/697) for US ENGLISH characters, then the application requester should set the default CHARNAME to ENGLISH. This is because CP/CS 37/697 corresponds to the CCSID of 37 which corresponds to the CHARNAME of ENGLISH.

The application requester default CCSID is the value of CCSIDMIXED if it is not zero; otherwise, it is the value of CCSIDSBCS. The application requester default CCSID is used for the following:

- The CCSID of SQL statements coded at the application requester
- The CCSID of host variables which represent character data
- The CCSID of character values described by an input or output SQLDA (when the SQLNAME field is *not* used to override the CCSID value)
- The CCSID of character data returned in a DESCRIBE SQLDA
- The CCSID of message tokens returned in an SQLCA

For more information on setting the default CHARNAME for an application requester, see "Setting the Application Requester Default CHARNAME and CCSIDs" on page 251. For more information on CCSIDs, see "CCSID Conversion" on page 245 and "Determining CCSID Values" on page 248.

Preparing for Database Regeneration

If the SYS0001 dbspace ever becomes too small to hold the catalog tables, or if the contents of the directory data set or a dbextent data set are damaged or destroyed and you do not have archives to restore them, the database can no longer serve your needs and must be regenerated.

The size and complexity of the regeneration task depends on the size and complexity of the database. This task includes:

- Regenerating the database, including any dbspaces, dbextents, and VM minidisks that may have been added since the previous generation
- Using the DBS utility to unload and reload all the data in the database, including the ISQL routines and the ISQL stored queries.
- Repreprocessing all application program packages
- Reestablishing the entire authority scheme
- Recreating all views and indexes.

One way to simplify this task is to keep a record of the various types of information you would need to reestablish the operating environment that existed in the previous database. In particular:

- Keep all the ACQUIRE DBSPACE, CREATE TABLE, ALTER TABLE, GRANT, CREATE INDEX, CREATE VIEW, and CREATE SYNONYM statements for the database in DBS utility job streams. These job streams can be run easily on the regenerated database.

  Note: If these statements are not kept, you can reconstruct them from information available in the system catalog tables. However, this could take a long time for a large production database.
- Keep all the VSAM Access Method Services statements used to define the VSAM data sets for the database, both for initial generation and for later ADD DBEXTENT operations. Also keep the statements used for any log reconfigurations. These statements can be used in one job to redefine the VSAM data sets.
- Keep all the input control statements for any ADD DBSPACE or ADD DBEXTENT operations. These statements can be used as input to the job that regenerates the database.
- Keep the database job control (DLBL, TLBL, and LIBDEF statements) up to date in a cataloged procedure. This procedure can be used by the job that regenerates the database.
- Keep the jobs used to preprocess each application program so that they can be run on the regenerated database (as separate jobs).

Database Generation Worksheet

This section provides two worksheets. Figure 13 covers the items that you must address in order to define the VSAM data sets for your database, and Table 4 covers the database generation control statements. Fill them out as you design your database; then refer to them when you do the actual database generation.

```
Need to define VSAM Master Catalog:  NO: __
                                    YES: __  CYL./BLOCKS: ____ VOLUME: _________
                                             ORIGIN: ____
Need to define VSAM User Catalog:    NO: __
                                    YES: __  CYL./BLOCKS: ____ VOLUME: _________
                                             ORIGIN: ____
Need to password protect data sets:  NO: __
                                    YES: __  Level: ______ Password: ___________


(Only one VSAM Data Space Required)
Data Space 1:   ORIGIN: ____                 CYL./BLOCKS: ___  VOLUME: _________

Data Space 2:   ORIGIN: ____                 CYL./BLOCKS: ___  VOLUME: _________

Data Space 3:   ORIGIN: ____                 CYL./BLOCKS: ___  VOLUME: _________

Data Space n:   ORIGIN: ____                 CYL./BLOCKS: ___  VOLUME: _________

Directory:      NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________


(Only one Log Required)
LOGDSK1:    NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

LOGDSK2:    NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

ALTLGD1:   NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

ALTLGD2:   NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

(Only one Dbextent Required)
Dbextent 1:     NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

Dbextent 2:     NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

Dbextent 3:     NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________

Dbextent n:     NAME: _____________________  CYL./BLOCKS: ___  VOLUME: __________
```

Figure 13. Important Factors for Installing Your Own Database

Table 4. Database Generation Worksheet

```
Database Name
   Server Name  ______________________________



   APPLID       ______________________________
```

```
Database Capacity Parameters:

 CUREXTNT _____________ (A value from 1 to 999 must be specified.)

 MAXPOOLS _____________ (Default is 32. Value can be from 1 to 999.)

 MAXEXTNT _____________ (Default is 64. Value can be from 1 to 999.)

 MAXDBSPC _____________ (Default is 1 000. Value can be up to 32 000.)
```

Table 4. Database Generation Worksheet (continued)

```
Nonrecoverable Storage Pools:

 POOL ________ NOLOG  (Storage pool 1 cannot be specified.)

 POOL ________ NOLOG

 POOL ________ NOLOG

 POOL ________ NOLOG
```

```
Database Extent (Dbextent) Placement:

 Dbextent      Storage Pool
 Number       (Default is 1)
 --------     --------------
   1              ___

   2              ___

   3              ___

   4              ___
```

Note: The number of dbextents must equal CUREXTNT, but one is required. The MAXEXTNT value determines the maximum number of database extents.

```
Public Dbspaces:

                                          SIZE         Storage Pool
 Purpose                             (In 4K Pages)    (Default is 1)
---------------------------------    -------------    --------------
Catalog Tables                            ____              1

Packages                                  ____             ___

HELP Text                                 ____             ___

ISQL                                      1024 (minimum)   ___

Sample Tables                              512 (minimum)   ___

_________________________________         ____             ___

_________________________________         ____             ___

_________________________________         ____             ___

_________________________________         ____             ___

_________________________________         ____             ___
```

Note: The public dbspaces for the catalog tables, packages, HELP text, ISQL, and the sample tables are required. The catalog tables must be in storage pool 1.

Table 4. Database Generation Worksheet (continued)

```
Private Dbspaces:

                                        SIZE          Storage Pool
 Purpose                            (In 4K Pages)    (Default is 1)
----------------------------------  -------------    --------------

__________________________________    ________         ________

__________________________________    ________         ________

__________________________________    ________         ________

__________________________________    ________         ________

__________________________________    ________         ________

__________________________________    ________         ________
```

```
Internal Dbspaces:

 Number: _____    Size in 4K Pages: ____    Storage Pool: ____
```

Note: The MAXDBSPC value determines the maximum total number of public, private, and internal dbspaces possible.

Chapter 3. Planning for Database Migration

If your installation already has a previous release of the database manager installed, you must consider the effect that migration to the new release will have on your existing databases and applications.

You can migrate to a DB2 Server for VSE Version 7 Release 5 database from:
- Version 7 Release 4
- Version 7 Release 3
- Version 7 Release 2
- Version 7 Release 1
- Version 6 Release 1
- Version 5 Release 1
- Version 3 Release 5
- Version 3 Release 4
- Version 3 Release 2
- Version 3 Release 1

Note: If you are on an earlier release, you will have to migrate to Version 3 Release 5 first and then to Version 7 Release 5.

This chapter also contains a section on:
- Release coexistence considerations

  It can be impractical to migrate all the databases in a local or distributed environment to the current level at the same time. For information on the level of coexistence that is possible see "Release Coexistence Considerations" on page 46.

  If you will be migrating databases on a VSE system to a VM system, see the *DB2 Server for VM System Administration* manual.

Migration Considerations

For users of an earlier version of the database manager, installing Version 7 Release 5 means loading the new code by running one or more IBM supplied programs, and migrating any existing databases. This section highlights the considerations that you should be aware of when doing this.

The topics are grouped by the release level of the database that is being migrated. Start at your release level and read to the end of this chapter. For example, if your database is Version 3 Release 1, you must review all the topics; if it is Version 3 Release 2, you need only read from that topic to the end of the chapter.

Increasing the HELPTEXT Dbspace

A database that is migrated keeps its existing HELPTEXT dbspace, which may not be large enough to support the Version 7 Release 5 HELP text. The size required for this dbspace depends on the number of national languages for which you have HELP text. It should be:

```
2,304 pages x number of languages installed.
```

This dbspace can be increased at any time before you install the current HELP text. For information, see the *DB2 Server for VSE & VM Database Administration* manual.

Migrating from Version 3 Release 1

Considerations for Invalid Indexes

Before you migrate, at least four dbspace blocks must be available in the database directory to allow for expansion of the invalid entities table. During migration, any entries in the invalid entities table are migrated to the new format. The new table format requires additional space in the directory. If there are any entries in the invalid entities table, it is possible that there may not be enough room in the directory to allow the table to be modified during migration.

For information about directory space verification, see the *DB2 Server for VSE Program Directory*.

Conversion of Packages

After migration, all packages are dynamically repreprocessed on first use. This conversion can cause a performance degradation over the first few days as the packages are referenced and repreprocessed.

To help minimize this degradation, the REBIND PACKAGE command is provided so that all packages can be recreated, if desired, after migration but before production. For information about this command, see the *DB2 Server for VSE & VM Database Services Utility* manual.

Migrating from Version 3 Release 2

Choosing a Server Name

With Version 3 Release 4 and later, you can specify a server name of up to 18 characters. See "Choosing an Application Server Name" on page 23.

Elimination of the SET XPCC Command

Before Version 3 Release 4 an application program could only access one database. In Version 3 Release 4 and later, up to 36 application servers can be active at the same time in your VSE system.

To make use of this new facility, you must remove the SET XPCC command and use the SET APPCVM command. For details on this command, see Chapter 5, "Operating the Online Support ," on page 81.

However, if you continue to access only one database, the SET XPCC command can still be used.

Choosing an Application Server Default CHARNAME

After migration, the database manager sets the application server default CHARNAME to ENGLISH, and sets the application server CCSID values as follows:
- CCSIDSBCS = 37
- CCSIDMIXED = 0
- CCSIDGRAPHIC = 0.

You can change the value of the default CHARNAME, which in turn determines the values for the three application server default CCSIDs. These four values are stored in the VALUE column of the SYSTEM.SYSOPTIONS catalog table. The corresponding values in the SQLOPTION column for these defaults are CHARNAME, CCSIDSBCS, CCSIDMIXED, and CCSIDGRAPHIC.

The value you choose for the default CHARNAME should accurately reflect the type of data that will be stored in the database: that is, the type of code page and character set that describes the data, and whether or not the database manager is to support DBCS characters or MBCS characters, or both. For more information, see "Character Set Considerations at Startup" on page 51, "Determining CCSID Values" on page 248, and "CCSID Conversion" on page 245. For a summary of the considerations for changing these values, see "Considerations when changing default CHARNAME and CCSID" on page 232.

Setting Migration CCSID Values

After choosing your default CHARNAME, you must also set your CCSID values for character and graphic data that existed before the migration to Version 3 Release 4. The CCSID value of character and graphic data stored in tables that were created before Version 3 Release 4 are specified by the three other rows (with SQLOPTION value MCCSIDSBCS, MCCSIDMIXED and MCCSIDGRAPHIC) in the SYSTEM.SYSOPTIONS catalog table. The migration CCSID values (MCCSIDSBCS, MCCSIDMIXED, and MCCSIDGRAPHIC) are used for single byte, mixed, and graphic data that was created prior to Version 3 Release 4 and therefore does not have a CCSID associated with it. The database manager sets the migration CCSID values as follows:
- MCCSIDSBCS = 37
- MCCSIDMIXED = 0
- MCCSIDGRAPHIC = 0.

If the code page and character set used to create the migrated data (that is, the data that was inserted into the database prior to Version 3 Release 4) is not CP/CS 37/697, these settings are not correct for your installation and **must** be changed. You can determine the CCSIDs for migrated data from the code page and character set that was used to generate the terminal controller where the data was entered.

For an example of how your choice of migration CCSID value affects the characters displayed, refer to page 250.

To determine if your database contains graphic or mixed data, issue the following query:

```
SELECT COUNT(*) FROM SYSTEM.SYSCOLUMNS
WHERE COLTYPE = 'GRAPHIC'  OR
      COLTYPE = 'VARGRAPH' OR
      COLTYPE = 'LONGVARG' OR
      SUBTYPE = 'M'
```

If the query returns a result of zero rows, the database contains neither graphic nor mixed data; a nonzero result indicates the number of columns in your database that do contain such data.

Handling SBCS Data: If your database contains only SBCS data (that is, the above query returns a result of zero) prior to Version 3 Release 4, the migrated CCSID values for mixed and graphic data (MCCSIDMIXED and MCCSIDGRAPHIC) must remain 0.

If the MCCSIDSBCS value of 37 is not correct for your installation, this must be changed to correspond to the code page and character set used to create the migrated data. For example, if the data was created with CP/CS 273/697

(GERMAN), the CCSID value you should use is 273. For a list of some of the SBCS CCSIDs and their character set and code page values, see Table 21 on page 247.

The row that you must update for data in tables created before Version 3 Release 4 is:

- SQLOPTION='MCCSIDSBCS'

  Change the value in the VALUE column to the appropriate SBCS CCSID (for example, 273 for GERMAN). The following statements show how to update or insert the row using this value:

```
UPDATE SYSTEM.SYSOPTIONS SET VALUE = '273'
WHERE SQLOPTION = 'MCCSIDSBCS'
```

```
INSERT INTO SYSTEM.SYSOPTIONS VALUES
('MCCSIDSBCS', '273',
'DEFAULT CCSID FOR MIGRATED SBCS CHARACTER COLUMNS')
```

Handling Mixed Data: If your database contains graphic or mixed data prior to Version 3 Release 4, you must update the VALUE column of SYSTEM.SYSOPTIONS for the row where SQLOPTION='MCCSIDMIXED' with the appropriate nonzero CCSID value. You **must** also update the row where SQLOPTION='MCCSIDSBCS' to the value of the SBCS component of the mixed CCSID, and the row where SQLOPTION='MCCSIDGRAPHIC' to the value of the DBCS component of the mixed CCSID. If these CCSIDs do not correspond to the components of the mixed CCSID, the wrong conversion selection tables are being used. For a list of some of the mixed CCSIDs and their component SBCS and DBCS CCSIDs, see Table 21 on page 247.

The rows that you must update for data in tables created before Version 3 Release 4 are:

- SQLOPTION='MCCSIDMIXED'

  Change the value in the VALUE column to the appropriate mixed CCSID. If you used DBCS characters before Version 3 Release 4, specify the appropriate CCSID value. For example, if you used Kanji characters, specify the value 5035. The following statements show how to update or insert the row using this value:

```
UPDATE SYSTEM.SYSOPTIONS SET VALUE = '5035'
WHERE SQLOPTION = 'MCCSIDMIXED'
```

```
INSERT INTO SYSTEM.SYSOPTIONS VALUES
('MCCSIDMIXED', '5035',
'DEFAULT CCSID FOR MIGRATED MIXED CHARACTER COLUMNS')
```

- SQLOPTION='MCCSIDSBCS'

  Change the value in the VALUE column to the appropriate SBCS CCSID. If you used DBCS characters before Version 3 Release 4, you must specify the SBCS component CCSID of the MCCSIDMIXED value. For example, if MCCSIDMIXED is set to 5035, specify 1027. The following statements show how to update or insert the row using this value:

```
UPDATE SYSTEM.SYSOPTIONS SET VALUE = '1027'
WHERE SQLOPTION = 'MCCSIDSBCS'
```

```
INSERT INTO SYSTEM.SYSOPTIONS VALUES
('MCCSIDSBCS', '1027',
'DEFAULT CCSID FOR MIGRATED SBCS CHARACTER COLUMNS')
```

- SQLOPTION='MCCSIDGRAPHIC'

  Change the value in the VALUE column to the appropriate graphic CCSID. If you used DBCS characters before Version 3 Release 4, this value must be the DBCS component CCSID of the MCCSIDMIXED value that you used. For example, if you used Kanji characters, specify 4396. The following statements show how to update or insert the row using this value:

```
UPDATE SYSTEM.SYSOPTIONS SET VALUE = '4396'
WHERE SQLOPTION = 'MCCSIDGRAPHIC'
```

```
INSERT INTO SYSTEM.SYSOPTIONS VALUES
('MCCSIDGRAPHIC', '4396',
'DEFAULT CCSID FOR MIGRATED GRAPHIC COLUMNS')
```

Considerations for Mixed Primary Keys with Field Procedures

If you are migrating from Version 3 Release 1 or Version 3 Release 2, the value of CCSID in SYSTEM.SYSKEYCOLS is NULL. For some primary keys, this value is not correct. In this case, you should drop and recreate the primary keys, which you can identify by running the ARIS341D procedure after migrating. (For information on this procedure, see the *DB2 Server for VSE Program Directory* manual.)

Considerations for EXPLAIN Tables

Several changes and enhancements were made to the EXPLAIN tables in Version 3 Release 4. If you have existing EXPLAIN tables they must either be renamed, or, dropped and recreated before using the EXPLAIN statement.

An IBM-supplied macro, ARISEXP, recreates the EXPLAIN tables for you.

For additional information on using EXPLAIN tables, see the *DB2 Server for VSE & VM Performance Tuning Handbook* manual.

Considerations for VSE Guest Sharing

VSE batch applications can access an application server on VM that is either remote or local. If the application server is in a remote network, the SET XPCC TARGET SYSARI command in the VSE IPL procedure must be replaced by the SET APPCVM TARGET command. If the application server is local, the SET XPCC TARGET SYSARI command in the VSE IPL procedure is not needed, and can be deleted.

Regardless of whether the application server is remote or local, an entry in the DBNAME directory may also be necessary to map the DBNAME to the resid when the DBNAME is greater than 8 characters, or when the DBNAME and the resid are different. For more information on the DBNAME directory, see “Setting Up the DBNAME Directory” on page 23.

Migrating from Version 3 Release 4

Considerations for Assembler Even Precision Packed Decimal

Prior to Version 3 Release 5, assembler host variables declared as even precision packed decimal were converted to odd precision by the preprocessor. As of Version 3 Release 5, the database manager supports assembler host variables defined as even precision packed decimal, and they are not converted to odd precision. In some cases, the lack of conversion may cause a datatype mismatch between a host variable and a column. To prevent potential performance degradation, applications affected by this change should be modified so the datatypes of the host variables exactly match the datatypes of the columns to which they will be compared.

Considerations for SQLSTATE Changes for SQL92 Support

The SQLSTATEs returned by several conditions were changed to comply with SQL92. Application programs that have a dependency on the SQLSTATE returned may be affected by these changes. See *DB2 Server for VM Messages and Codes* for information on the changed SQLSTATEs.

Migrating from Version 3 Release 5

Considerations for Uncommitted Read

Prior to Version 5 Release 1, the database manager accepted isolation level uncommitted read as a preprocessor parameter, but internally the isolation level was escalated. As of Version 5 Release 1, isolation level uncommitted read is fully supported. However, this isolation level to take effect, packages that were prepped with uncommitted read in a previous release must be explicitly repreprocessed after migration.

Considerations for Support of ESA-mode Processors Only

Any user exits (date, time, or accounting), field procedures, or applications that run in single user mode that are dependent on running in a 370 mode virtual machine must be converted to execute in an ESA mode virtual machine. AMODE 24 is still supported, so exits, field procedures, and single user mode applications that require AMODE 24 are not affected.

Considerations for the Renaming of the Product

The text of several messages was modified as the result of the renaming of the product. Applications with dependencies on the text of messages may be affected.

Considerations for the Removal of the User Facility Subset

The User Facility Subset is no longer supported; machines on which the subset was previously installed must now contain the full product.

Migrating from Version 5 Release 1

Choosing the Default CHARNAME for All Application Requesters

After migration, the application requester default CHARNAME is determined from the SQLGLOB file. By default it is set to INTERNATIONAL, and the application requester CCSID values are as follows:
- CCSIDSBCS = 500
- CCSIDMIXED = 0
- CCSIDGRAPHIC = 0.

To ensure the integrity of character data representation and to reduce the performance overhead associated with CCSID conversion, it is important to choose the appropriate CHARNAME for the code page used by each application requester. See “Choosing the Default CHARNAME and CCSID for Application Requesters” on page 34 and “Setting the Application Requester Default CHARNAME and CCSIDs” on page 251. For more general information on CCSIDs, see “CCSID Conversion” on page 245 and “Determining CCSID Values” on page 248.

Considerations for VSE DRDA Online Requester Support

The format of the DBNAME directory source file member, ARISDIRD, has changed and must be modified and a new phase created. See “Setting Up the DBNAME Directory” on page 23.

Considerations for RDS Above 16M

After migration, the RDS component will be loaded above 16M whenever possible. For information, see “V6R1 and V5R1 Incompatibilities” on page 421.

Migrating from Version 6 Release 1

Considerations for the DBNAME Directory

The format of the DBNAME directory has changed. As a result, you must modify the DBNAME directory source file member (ARISDIRD), and create a new phase. See “V7R1 and V6R1 Incompatibilities” on page 421 and “Setting Up the DBNAME Directory” on page 23, for more information.

Considerations for Key Enablement

When running VSE/ESA 2.5 or later, DB2 Server for VSE is key-enabled. For information on setting up the DB2 key, see the *DB2 Server for VSE Program Directory*.

Migrating from Version 7 Release 1

There are no issues to consider when migrating from Version 7 Release 1 to Version 7 Release 5.

Migrating from Version 7 Release 2

There are no issues to consider when migrating from Version 7 Release 2 to Version 7 Release 5.

Migrating from Version 7 Release 3

There are no issues to consider when migrating from Version 7 Release 3 to Version 7 Release 5.

Migrating from Version 7 Release 4

There are no issues to consider when migrating from Version 7 Release 4 to Version 7 Release 5.

Release Coexistence Considerations

For installations with multiple databases, you should migrate all your databases to the current level. All users have the same features available to them, and future database migrations are easier.

Applications at any supported release level can access application servers at any supported release level. However, if an application requester and application server are at different release levels, any functions used must be available in both release levels. That is, you cannot use any new release facilities from ISQL, DBS Utility, or application programs when the application server is running a different level of DB2 Server for VSE & VM than the application requester.

All existing applications that accessed a database before the database was migrated to another release level continue to work after migration.

See Appendix I, "Incompatibilities Between Releases," on page 389 for incompatibilities that exist between each release and the next release.

Changing the Server Name and Application Server Identifier

Situations exist where you may want to change the application server name (DBNAME), application identifier (APPLID), or CICS transaction program name (TPN).

These changes are made in the DBNAME directory. For details, see "Setting Up the DBNAME Directory" on page 23.

Moving a Database

Using the SQLDBDEF Utility

The SQLDBDEF can be helpful if you are moving your database. This utility extracts the definition of database objects from a DB2 Server for VSE & VM database, and generates a DBSU job that can be used to create the same objects on another DB2 database. The target database can be any DB2 database, for example, DB2 Server for VSE & VM, DB2 UDB for OS/390, DB2 UDB for Linux, etc. Once the objects have been created on the target platform, the load utilities of the target database can be used to load the data. Packages can be unloaded from the source database and reloaded to the target database so that existing client applications can continue to be used.

For more information on the SQLDBDEF utility, see Appendix G, "Service and Maintenance Utilities," on page 381.

Chapter 4. Planning for Operation of the Database Manager

Once the DB2 Server for VSE code is installed and your database generated, the operator can start the application server so that users can access the databases and submit SQL statements. This chapter explains the planning tasks associated with starting, running, and stopping the application server. For information on the actual operator commands, see the *DB2 Server for VSE & VM Operation* manual.

The examples in this book assume that you have loaded the IBM-supplied procedures containing the job control statements needed for each database and for referencing the DB2 Server for VSE library.

Starting the Application Server

This section discusses the following topics:
- Modes of operation
- Multiple user mode initialization parameters
- Single user mode initialization parameters
- Tape support
- Starting the application server in multiple user mode
- Running multiple user mode applications
- Starting the application server in single user mode
- Overriding initialization parameters
- Creating a parameter data set

Modes of Operation

The database manager can be operated in either multiple user mode or single user mode.

In *multiple user mode*, more than one user or application can concurrently access the same database. The database manager runs in one VSE system partition while applications run in other partitions. The initialization parameter SYSMODE=M defines this mode.

In *single user mode*, only one user or application can be run at one time. Both the database manager and the application program run in the same VSE system partition. The initialization parameter SYSMODE=S defines this mode.

Many of the database manager facilities, including support for ISQL, CICS transactions, and VSE/ICCF, are available only in multiple user mode. Support for the DBS utility, preprocessors, and batch applications is available in both multiple and single user modes. Support for special facilities (such as ADD DBEXTENT) is available only in single user mode.

Multiple User Mode Initialization Parameters

Table 5 on page 48 identifies the initialization parameters that apply when the database manager is operating in multiple user mode, and lists their defaults. A discussion of the appropriate settings for these parameters follows.

Table 5. Multiple User Mode Initialization Parameters

Parameter	Default	Minimum	Maximum
Environment Parameters			
DBNAME=name	From DBNAME directory	—	—
RMTUSERS=nnnnn	0	0	65535
SYSMODE=M	M	—	—
STARTUP=W|R|F|U	W	—	—
PARMID=name	None	—	—
DBPSWD=password	None	—	—
CHARNAME=name	INTERNATIONAL	—	—
ACCOUNT=T|D|E|N	N	—	—
SYNCPNT=Y|N	If RMTUSERS > 0, Y	—	—
DSPSTATS=nn	0	0	21
TCPDISPB	1	1	100
TCPMAXRT=n	158	1	9999
TCPPORT=n	DBNAME Directory	0	65535
TCPRETRY=Y|N	Y	—	—
SECALVER=Y|N	N	—	—
SECTYPE=DB2|ESM	DB2	—	—
Performance Parameters			
NCUSERS=n	5	1	251
NPACKAGE=n	10	1	32766
NPACKPCT=n	30	0	100
NPAGBUF=n	10 + NCUSERS x 4	10	40000
NDIRBUF=n	NPAGBUF	10	40000
NLRBU=n	1000	10	583333
NLRBS=n	(2 x NCUSERS) + (NLRBU x NCUSERS)/2 +10	larger of 50 or (2 x NCUSERS)	
DISPBIAS=n	7	1	10
NCSCANS=n	30	1	655
LTIMEOUT=n	0	0	99999
PTIMEOUT=n	180	0	99999
PROCMXAB=n	0	0	255
Recovery Parameters			
LOGMODE=Y|A|L	Y	—	—
CHKINTVL=n	10	1	99999999
SLOGCUSH=n	90	11	90
ARCHPCT=n	80	10	99
TAPEMGR=N|Y	N	—	—
ARCHTAPE=REW|UNL	REW	—	—
SOSLEVEL=n	10	1	100

Table 5. Multiple User Mode Initialization Parameters (continued)

Parameter	Default	Minimum	Maximum
Service Parameters			
DSPLYDEV=L|C|B	L	—	—
DUMPTYPE=P|F|N	F	—	—
EXTEND=Y|N	N	—	—
TRACDBSS=nnn...	000...	000...	222...
TRACRDS=nnnnnnn	0000000	0000000	2222222
TRACWUM=n	0	0	2
TRACDRRM=nnnn	0000	0000	2222
TRACDSC=nn	00	00	22
TRACCONV=n	0	0	2
TRACSTG=n	0	0	1
TRACEBUF=n	0	0	99999

Environment Parameters

DBNAME

This parameter specifies the application server to be started. Different application servers can be started in different partitions (up to a maximum of 36). With multiple server support, different procedures can be written to link the appropriate disks for each database. When starting the application server, correlate the procedure name with this parameter. If not specified, the application server is started with the default specified in the DBNAME directory. For more details, see "Choosing an Application Server Name" on page 23.

The following examples show the DBNAME parameter specified correctly with two procedures to start two different application servers.

```
// JOB xxxxx
// EXEC PROC=ARIS75SL
// EXEC PROC=SQLDB1
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='STARTUP=W,DBNAME=SQLDB1_NEWYORK_INV'
/*
/&


// JOB yyyyy
// EXEC PROC=ARIS75SL
// EXEC PROC=SQLDB36
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='STARTUP=W,DBNAME=SQLDB36_TORONTO_INV'
/*
/&
```

Figure 14. Examples of Job Control to Start a Database in Multiple User Mode

RMTUSERS

This parameter can be specified only if the DRDA code can be installed.

The value specified for RMTUSERS is the maximum number of remote users that can access the application server, and can be set to any number from 1 to 65535 (inclusive). The default is 0. The appropriate value for RMTUSERS depends on the availability of virtual storage.

If RMTUSERS is not specified or the value is 0, remote users will not be able to access the application server.

SYSMODE

This parameter is used to specify either single(S) or multiple(M) user mode. Set it to M to initialize the database manager for multiple user mode operation. This is the default mode.

STARTUP

This parameter specifies how the database will be started:

- Most of the time let STARTUP default to W (warm start).
- Use STARTUP=R (restore) to restart the application server and restore the database from an archive tape file. This setting causes the VSAM data sets to be reset before the data is restored.
- STARTUP=F (fast restore) also restores the database from an archive tape, but does not format the data sets. Only use it if you have not changed the database files. (The database files would be changed, for example, when there is a media failure.)
- Specify STARTUP=U (user restore) if you have archived and restored the database with user facilities.

For more information, see “Restoring the Database” on page 158.

PARMID

This parameter can be used to specify an A-type source member containing the values for the other initialization parameters. Application program parameters (user parameters) cannot be included. Figure 15 shows an example of startup of the application server that uses the PARMID initialization parameter.

DBPSWD

When you define the data sets for your database, you can define a password to protect them from unintentional or malicious access. All the data sets for a database must have the same VSAM password, which is defined with the VSE/VSAM DEFINE CLUSTER command. Then, when starting the application server, the operator must specify this password with the DBPSWD parameter, as shown in Figure 15. If the specified password does not match the one defined for the data sets, the operator is prompted to supply the correct one.

```
// JOB MULTI
// EXEC PROC=ARIS75SL
// EXEC PROC=ARIS75DB
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='PARMID=WARM1,DBPSWD=password'
/*
/&
```

Figure 15. Job Control to Start in Multiple User Mode with Password-Protected Data Sets

CHARNAME

This section discusses the following:

- Character set considerations at startup
- National language considerations at startup.

Character Set Considerations at Startup: Use the CHARNAME parameter to specify the CCSIDs to be used as the application server defaults. The default CCSIDs determine the character sets and code pages to be used to interpret statements and return results.

The valid CHARNAME values you can specify are ENGLISH (CCSID=37), INTERNATIONAL (CCSID=500), and all the values that are in the CHARNAME column of the SYSTEM.SYSCCSIDS catalog table.

The database manager obtains the CCSIDs associated with the CHARNAME by looking up the row of the SYSTEM.SYSCCSIDS catalog table where the CHARNAME column matches the CHARNAME parameter. It also obtains the classification and translation tables associated with the CHARNAME by looking up the row of the SYSTEM.SYSCHARSETS catalog table where the NAME column matches the CHARNAME parameter. The classification table is used to identify valid characters in identifiers. The translation table is used to indicate how to fold ordinary lowercase identifiers to uppercase.

For CHARNAMEs ENGLISH and INTERNATIONAL, their CCSID values, the classification table and the translation table are stored internally. The rows in SYSTEM.SYSCCSIDS and SYSTEM.SYSCHARSETS for these CHARNAMEs are for reference purposes only and are not used by the database manager.

During startup, if you do not specify the CHARNAME parameter, the application server uses the same CHARNAME that was used the last time it was started. The values stored in the rows where SQLOPTION equals CHARNAME, CCSIDSBCS, CCSIDMIXED, and CCSIDGRAPHIC are for reference purposes only. They reflect the current values associated with the system. The only way to change the default values is by starting the application server with a different CHARNAME parameter. Any updates to the values in the SYSTEM.SYSOPTIONS table are ignored during startup.

Note: The database manager determines the current default CHARNAME from the CCSID attribute of the CNAME character column in the SYSTEM.SYSCOLUMNS catalog table. If this value is null, then 37 is used (a CCSID of 37 corresponds to a CHARNAME of ENGLISH). The database manager uses the CCSID value to locate the corresponding row in the SYSTEM.SYSCCSIDS catalog table to obtain the associated CHARNAME. The value in the CHARNAME column of this row is the current application server default CHARNAME.

When you specify a value for the CHARNAME parameter that is different from the current application server default CHARNAME, you are prompted to choose whether or not you want to change the application server default CHARNAME. If you specify YES and have supplied a valid CHARNAME value, the database manager updates the application server default values for CHARNAME, CCSIDSBCS, CCSIDMIXED, and CCSIDGRAPHIC. It also modifies the CCSID attribute of all character columns that are part of the catalog tables to the application server default CCSID. The CCSID attribute of character columns that are not part of the catalog tables are not modified. If the value for CCSIDMIXED is not zero, this value is used as the application server default CCSID. If the value for CCSIDMIXED is zero, then the application server default CCSID is the value of CCSIDSBCS.

Note that the tables which have their CCSID modified when the CHARNAME is changed include:

- All tables created by SYSTEM
- The following tables created by SQLDBA:
  - SQLDBA.ROUTINE
  - SQLDBA.STORED QUERIES
  - SQLDBA.SYSLANGUAGE
  - SQLDBA.SYSTEXT2
  - SQLDBA.SYSUSERLIST

When a CHARNAME is changed, the following should be considered:

1. The FIPS Flagger package **must** be reloaded by using the ARIS360D procedure. Failure to do this can cause SQLCODE=-931 (SQLSTATE=58004). This will render the agent reporting the SQLCODE error unable to preprocess packages until the application server is started. Once the FIPS Flagger package is reloaded or repreprocessed, this error will not occur.
2. All views which are dependent on the tables that had their CCSID modified must be dropped and recreated.

   The following query lists all such view packages:
   ```
     SELECT CREATOR, TNAME, PLABEL
       FROM SYSTEM.SYSACCESS
       WHERE TABTYPE  = 'V'
       AND VALID = 'N'
   ```
   This query is useful in that owners of affected views can be notified to drop and recreate their view before they try and use the view and get an error (SQLCODE=-835, SQLSTATE=56049, with SQLERRD1 set to -833).
3. All packages which are dependent on the tables that had their CCSID modified must be dropped and recreated.

   The following query lists all such packages:
   ```
     SELECT CREATOR, TNAME, PLABEL
       FROM SYSTEM.SYSACCESS
       WHERE TABTYPE  = 'X'
       AND VALID = 'N'
   ```
   This query is useful in that owners of affected packages can be notified to rebind the packages instead of having them dynamically repreprocessed at run time. The DBS utility REBIND PACKAGE command can be used to rebind the packages listed.
4. The ISQL package (SQLDBA.ARIISQL) and DBS utility package (SQLDBA.ARIDSQL) can be reloaded and recreated using the ARIS360D procedure. If this is not done, the first time these packages are used, they will be dynamically repreprocessed.

To check if all the above activities have been done, run the following query:
```
  SELECT CREATOR, TNAME, PLABEL
    FROM SYSTEM.SYSACCESS
    WHERE VALID = 'N'
```

If there are no rows found, all packages have been either recreated, reloaded, rebound or dynamically repreprocessed and the VALID column value for the package in SYSTEM.SYSACCESS has been changed to “Y”.

Note that CCSID conversion of the data in catalog tables does not occur: only the CCSID attribute of the columns is modified. If you change the application server default CHARNAME, system objects of the character data type (for example, table names and column names) stored in the catalog may be displayed differently. The reason for this is that a code point may represent different characters in different code pages.

If you want to change the application server default CHARNAME, the default will not be changed if:

- You specify an invalid value for the CHARNAME parameter
- An error occurs in the verification of the
  - New CHARNAME CCSID values
  - Classification table
  - Translation table.

When the application server is started, it records the application server default values for CHARNAME, CCSIDSBCS, CCSIDMIXED, and CCSIDGRAPHIC in the SYSTEM.SYSOPTIONS catalog table. To obtain these values, you can query the table. For example, to determine the name of the character set that is currently in use, issue:

```
SELECT VALUE
  FROM SYSTEM.SYSOPTIONS
  WHERE SQLOPTION = 'CHARNAME'
```

For more information about character sets, see Chapter 12, “Choosing a National Language and Defining Character Sets,” on page 231.

National Language Considerations at Startup: You can use the SET LANGUAGE command from the operator console to choose a national language so that DB2 Server for VSE messages can be received in the selected language. (You cannot choose a double-byte character set (DBCS) language, as the VSE operator console does not support DBCS.) For more information see “National Language Support for Messages and HELP Text” on page 259.

ACCOUNT

This parameter enables the accounting facility. If ACCOUNT=T or ACCOUNT=D or ACCOUNT=E is specified, accounting records will be generated and directed to either a tape file, a SAM DASD file or a VSAM ESDS file, respectively. (The tape or DASD file must be identified in your job control for starting the application server.) If the default value of ACCOUNT=N is specified, accounting information is not generated.

For a complete description of the accounting facility, see Chapter 10, “Using the Accounting Facility,” on page 187.

SYNCPNT

This parameter specifies whether or not a sync point manager (SPM) will be used to coordinate DRDA2 DUOW two-phase commit and resynchronization activity. It is only meaningful when the RMTUSERS parameter is greater than zero.

If Y is specified, the server will use a sync point manager, if possible, to coordinate two-phase commits and resynchronization activity. If N is specified, the server will not use an SPM to perform two-phase commits. If N is specified, the database manager is limited to multi-read, single-write distributed units of work and it can be the single write site. If Y is specified, but the database manager finds that a sync point manager is *not* available, then the server will operate as if N was specified.

The default is SYNCPNT=Y, if RMTUSERS is greater than zero.

DSPSTATS

This two digit parameter specifies what information is displayed and what level of detail is displayed. If 0 is specified, nothing is displayed. If 1 is specified, the minimum information is displayed. If 2 is specified, more detail is displayed. The positional digits correspond to the following informational displays: the first is checkpoint performance information and the second is counter information to be displayed at system shutdown.

If the first option is 1, then format 1 of message ARI2052I is displayed every time a checkpoint occurs. This is useful in determining how often checkpoints occur. If the first option is 2, then format 2 of message ARI2052I is displayed every time a checkpoint occurs. This is useful in determining if checkpoint processing is causing a performance problem.

If the second option is 1, then the “COUNTER *” operator command is issued just before the application server is shutdown. This is useful for performance tuning.

The SET command changes the value of this parameter without having to stop and restart the application server. For more information on the SET operator command, see the *DB2 Server for VSE & VM Operation* manual.

SECALVER

This parameter determines if the application server will accept users that have already been verified by another system. If SECALVER=Y, verified users will be accepted. The requester only needs to send a user ID to be validated. If SECALVER=N, verified users will not be accepted. The requester must send a user ID and password to be verified.

Note: This parameter is only used when validating users are connecting via TCP/IP or when users send the ACCSEC and SECCHK DRDA datastreams in their connect request.

SECTYPE

This parameter determines if the application server will validate a user ID and password for connect authority using an external security manager or by checking the DB2 SYSUSERAUTH catalog table. If SECTYPE=ESM an external security manager will be used to validate the user ID and password. The external security manager must support the RACROUTE application programming interface. If SECTYPE=DB2, the user ID and password are validated by checking the SYSUSERAUTH catalog table.

Note: This parameter is only used when validating users are connecting via TCP/IP or when users send the ACCSEC and SECCHK DRDA datastreams in their connect request.

TCPDISPB

This value specifies how frequently the TCPIP agent is dispatched when there are no agents ready to run and a system wait must be done. The TCPIP agent will be dispatched every *n* times a system wait must be done. Otherwise, the system wait is performed by the DB2/VSE dispatcher.

Valid values are 1 to 100.

Setting the value to a high number can improve the performance of agents that are doing I/O intensive queries and are not using TCP/IP as the communications protocol. This setting will not affect the performance of CPU intensive queries as much as it will affect I/O intensive queries.

Setting the value to a high number can degrade the performance of users that are using TCP/IP as the communications protocol.

This value is ignored if RMTUSERS=0. It is only active if TCP/IP support is enabled.

TCPMAXRT

This parameter specifies the maximum number of times the application server will attempt to re-enable TCP/IP support if it was disabled.

For a complete description of TCP/IP support, see Chapter 15, “Using TCP/IP with DB2 Server for VSE,” on page 335.

TCPPORT

This parameter specifies the TCP/IP port number that the application server will use to listen for incoming TCP/IP connect requests.

If this parameter is not specified, TCP/IP support will be initialized and the DBNAME Directory on the TCP/IP client disk will be searched to determine the port number that the application server will use.

If this parameter is specified with a non-zero value, TCP/IP support will be initialized and the value specified will be used as the port number that the application server will use.

If this parameter is specified with a value of 0, TCP/IP support will not be initialized.

For a complete description of TCP/IP support, see Chapter 15, “Using TCP/IP with DB2 Server for VSE,” on page 335.

TCPRETRY

This parameter determines if the application server will automatically attempt to re-enable TCP/IP support if it becomes disabled.

For a complete description of TCP/IP support, see Chapter 15, “Using TCP/IP with DB2 Server for VSE,” on page 335.

Performance Parameters

NCUSERS

This parameter defines the maximum number of *real* agents that the database manager can actively handle at any one time, limiting the number of users that can be supported by the database manager. The value of NCUSERS is usually less than the number of connected users anticipated, because not all users will be accessing data at the same time. This value directly affects the size of the database partition required.

The number of NCUSERS is limited because some static agent storage for each real agent is obtained below 16 megabytes.

Figure 16 provides guidelines for setting the NCUSERS parameter. Because these are only guidelines, you should modify them to concur with the activity on your system. For additional information, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

```
NCUSERS=   1 for each 1-2 users of ISQL (or other query products)
         + 1 for each 2-5 application program developers (VSE/ICCF users)
         + 1 for each 4-10 non-ISQL CICS users
         + 1 for each batch partition supported
         + 1 for each 4-10 remote users
```

Figure 16. Guidelines for the NCUSERS Parameter

Each ISQL user can generate a high level of system activity. If you set NCUSERS so that all ISQL users can be active at the same time (NCUSERS=number of ISQL users), you minimize the time that any one user must wait for services. However, if this number is large, it may cause the database manager to be overloaded. To prevent this, you should also use the CICS DFHSIT parameter CMXT, which limits the number of users who can be logged on to ISQL. For information on this parameter, see “Controlling Access by ISQL Users” on page 119.

Application developers (VSE/ICCF users) typically do a considerable amount of other activity (such as VSE/ICCF file editing or output scanning). These users require less service from the database manager, so NCUSERS can be lowered accordingly.

If you are using VSE guest sharing, the NCUSERS of the VM database machine should be increased by the number required for the VSE guest. The demand for services from CICS transaction processing can vary widely, depending on the nature of the transactions.

The demand for services from batch application programs can also vary considerably. If you have online or interactive activity on the database manager, consider limiting the amount of concurrent SQL batch processing.

Note: When the application server is started, there may be one or more in-doubt logical units of work (LUWs). The value of NCUSERS must be large enough to handle these. When they have been resolved, the DB2 Server for VSE agent structures are used to handle new users. The creation and use of agent structures for resolving in-doubt LUWs takes precedence over all new user logical units of work. For more information about in-doubt LUWs, see “Resolving In-Doubt Transactions” on page 98.

NPACKAGE

This parameter defines the maximum number of packages in an LUW, and together with the value specified for NCUSERS, determines the size of the *package cache*. The size of the package cache limits the number of packages that can be present in storage simultaneously. (Package cache size = NPACKAGE x NCUSERS.) The default value of NPACKAGE is 10, and that for NCUSERS is 5, giving a default package cache of 50, allowing 50 packages to be present in storage simultaneously.

In general, increasing the size of the package cache improves performance of the database manager. However, do not increase it to the point where system paging becomes too great. For more information, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

NPACKPCT

This parameter defines the percentage of the package cache that is used in the calculation of the package cache *threshold*. The size of the threshold determines the number of loaded packages that are kept in storage at the end of an LUW. (Threshold = NPACKPCT percent of package cache.) If the threshold is exceeded, the loaded packages are freed and returned to the package cache.

The default values for NPACKPCT and the package cache are 30 and 50 respectively, giving a threshold of 15. In general, increasing the size of the threshold improves performance. For more information, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

NPAGBUF

This parameter specifies the number of 4096-byte data pages kept in storage buffers at one time. The number of data buffers you want depends on the number of active users and the nature of their request. The default for NPAGBUF assumes an average of four buffer pages for each potentially active user (NCUSERS x 4), plus ten buffer pages for the buffering of catalog and log information.

In general, increasing NPAGBUF improves the performance of the database manager. However, increasing it also requires an increase in the size of the database partition. Also -- and more importantly -- it can cause an increase in the paging rate of the system. It is more efficient to let the database manager do more I/O operations than it is to let the system do more paging; database I/O operations are overlapped whereas system paging operations are not. Therefore do not increase NPAGBUF to the point where system paging becomes too great.

For more information about NPAGBUF, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

NDIRBUF

This parameter determines the number of 512-byte directory pages to be kept in storage. Increasing it reduces the number of I/O operations. Again, bigger is better, until you either run out of virtual storage or cause too much system paging. Each directory page addresses 128 data pages.

When you set NPAGBUF and NDIRBUF, you have to choose how to split buffer space between data pages and directory pages. At least initially, you should set them to the same value. Issue the COUNTER commands to see the actual I/O activity; then adjust NPAGBUF and NDIRBUF.

For more information about NDIRBUF, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

NLRBU and NLRBS

NLRBU specifies the maximum number of lock request blocks allowed for one active user, while NLRBS specifies the number allowed for *all* active users. (Usually, two lock request blocks are used for every lock that a user holds.)

The database manager can perform lock escalations, increasing the granularity of data being locked from either row or page level to dbspace level. In general, you only need to change the default values of NLRBU and NLRBS if contention problems occur. Increasing them reduces the number of lock escalations performed by the database manager.

When either the NLRBU limit for a user is reached or the NLRBS limit is approached, lock escalation occurs. This results in fewer locks being required, and lock request blocks being freed. This in turn reduces the opportunities to share data. For example, when locking is done at a row level, many users may be updating the same dbspace at the same time. When it is escalated to the dbspace level, only one user can update rows in that dbspace. Everyone else must wait until that person's update is committed or rolled back.

Escalation can also cause deadlocks. A deadlock occurs when two or more LUWs are in wait states and dependent on the completion of LUWs that are also in wait states. For example, suppose two users are updating tables in a dbspace. When the lock size is escalated to a dbspace level, both users can be locked out, with each waiting for the other to complete an LUW. The database manager resolves situations like these by rolling back the newest LUW. For more about locking, see the *DB2 Server for VSE & VM Application Programming* manual.

If the default values for NCUSERS (5) and NLRBU (1000) are used, the database manager defines 2520 lock request blocks, each of which requires 24 bytes; 60480 bytes of virtual storage are required for lock request blocks. With these defaults, one application could use 1000 lock request blocks and four other applications could simultaneously use an average of 370 lock request blocks each, before causing an escalation.

Even though two lock request blocks are needed for each lock, the default values allow a large number of locks for each application. With the defaults, one application could use 500 locks while four other applications use an average of 185 locks each.

You should use the NLRBU and NLRBS default values at first, and increase them if users either are experiencing delays when they access the database manager, or if they are receiving SQLCODEs of -911, -912, or -915 (rollbacks that occur because of deadlock, insufficient lock request blocks for the database manager, or insufficient lock request blocks for a user application, respectively).

Note: These SQLCODEs may also be received during preprocessing, as the locks are required then as well.

To test the frequency of lock escalations and of deadlocks, use the COUNTER operator command. Specify both the ESCALATE and the LOCKLMT counters to get the number of successful escalations and the number of unsuccessful escalation attempts respectively. (An escalation can fail if the LUW that reached the lock limit is rolled back because of a deadlock, or if a sufficient number of lock request blocks cannot be freed.) For example, suppose the operator issues the command `COUNTER ESCALATE LOCKLMT` a few times a day and normally receives results in the range of 10 to 150 for ESCALATE, and 0 to 5 for LOCKLMT. If, one day, the results are 428 for ESCALATE and 23 for LOCKLMT, a locking problem would be indicated.

In addition, the `SHOW LOCK MATRIX` command can be used to display information about lock request block usage to determine whether unexpected delays are caused by locking; to monitor how the database manager is using lock request blocks; and to determine the lock request blocks required for a single application or for a run of a preprocessor.

One of the values displayed is called MAX USED BY LUW: the maximum number of lock request blocks used by any one application during an LUW. (When any LUW starts to exceed NLRBU and the escalation process occurs, MAX USED BY LUW is set to zero.) All this information can help you determine the required values for NLRBU and NLRBS.

To establish the lock request block requirements for running a preprocessor, or for an application that is causing contention problems:
1. Start the application server in multiple user mode with NCUSERS=1, NLRBU about five times its current setting, and NLRBS set to the same value as NLRBU.
2. Start the application and allow it to complete processing.
3. Verify that no escalation occurred by displaying the ESCALATE and LOCKLMT counters. If no escalation occurred, MAX USED BY LUW will show the number of lock request blocks required.
4. If an escalation did occur, set NLRBU to a value greater than or equal to MAX USED BY LUW, then start the application server again, and rerun the application.

If necessary, reset NLRBS. For example, suppose NLRBU is set to 1100, and two users will run their applications -- each requiring 1100 lock request blocks -- at the same time. Also assume that any other application requires about 500 lock request blocks. If NCUSERS is 5, then set NLRBS to at least 3700 (1100 for each of two applications and 500 for each of three additional applications).

If an application requires more lock request blocks than you have virtual storage for, you should consider the following alternatives:
- Use either the SQL *ALTER DBSPACE* or the SQL LOCK statement to change the locking level of the dbspace used by the application. The ALTER statement permanently changes the locking level for all applications, while the LOCK statement can be inserted into an application, and used to change the locking level only when that application runs. The LOCK statement is the preferred way to temporarily modify the locking level, because it involves no update to the catalog tables.
- Consider changing the application: perhaps it is holding locks longer than necessary. Additional SQL *COMMIT WORK* statements in the application may necessitate fewer locks.
- Consider running the application by itself: either in single user mode, where no locking is required, or in multiple user mode with a reduced NCUSERS and with NLRBU and NLRBS set as required.

For more information about locking problems and how to solve them, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

DISPBIAS

This parameter determines how the dispatcher selects the order in which agents get serviced by the database manager. To set it, you need to understand how the dispatcher works. Only one agent at a time can be serviced; the other agents wait in a queue. Within this queue, agents are prioritized according to their estimated resource consumption: those estimated to consume the least are placed at the top, while those estimated to consume the most are placed at the bottom.

When the active agent returns to the dispatcher, the next agent at the top of the queue is dispatched. Every time an agent is dispatched, the database manager reevaluates the priority of the remaining agents, and requeues them according to their new priorities.

A pure priority dispatcher can present some problems, however. If many short-running LUWs are present, the longer-running ones may never get serviced: they are always at the bottom of the queue. To avoid this problem, *fair-share auditing* is used, whereby all the agents in the queue are checked periodically to see if they are receiving adequate service. When one is found that is not, its priority is changed and it is moved to the top of the queue.

If fair-share auditing is done frequently, the dispatcher tends to operate more like a *round-robin* dispatcher: agents get equal service because those at the bottom of the queue get bumped to the top more frequently. If it is done infrequently, the dispatcher tends to operate more like a *priority* dispatcher: agents get prioritized service because long-running agents are forced to wait at the bottom of the queue longer. (Eventually, fair-share auditing causes these agents to get service.)

The DISPBIAS parameter determines how often fair-share auditing is done. When it is set low (near 1), fair-share auditing is done frequently, and the dispatcher operates more in round-robin mode. When it is set high (near 10), fair-share auditing is done infrequently, and the dispatcher operates more in priority mode.

Initially, you should use the DISPBIAS default of 7. If your long-running LUWs are getting poor service, you may want to use a lower value; if your ISQL users are often waiting for long-running applications to complete, you may want to use a higher value. You can use the SET operator command to change the value of DISPBIAS without having to stop and restart the application server. See the *DB2 Server for VSE & VM Operation* manual for more information on the SET operator command.

Note: Any changes you make using the SET command are only in effect while the application server is running. If you stop and restart the application server, it will use the settings you specified in the startup procedure.

You may be tempted to set DISPBIAS to 10 to get good response time for ISQL users. Keep in mind, however, that a long-running LUW can hold a large number of locks. If other users are waiting for those locks, they must wait until the application frees them. If the application is waiting at the bottom of the queue, everyone is waiting. In this situation, you would want to have fair-share auditing occur more frequently, so the long-running unit can free the resources it has locked. The default of seven represents a balance between the interests of long-running and short-running LUWs.

NCSCANS

This parameter determines the number of internal control scan blocks kept for accessing tables and indexes. These blocks can vary in size and number depending on the type of query being performed. This discussion is concerned with long-running requests that might be queries or database change operations.

Scan control blocks contain positioning information related to a query. The positioning information can result from a user-defined cursor or by an internal cursor created by RDS. If an index is involved in the query, the size of the scan control block depends on the key size for that index. An average scan control block is assumed to be 50 bytes (32 bytes for control information, and an average key length of 18 bytes).

The maximum table size to hold the scan control block entries for each agent is 32 kilobytes (32768 bytes). This can contain 655 entries of 50-byte scan control blocks, which in general, is enough to support 255 user-declared cursors. If, however, the key lengths for indexes are long, the scan table supports fewer user cursors. For example, if the key length for a given index associated with a cursor is 255 bytes, an entry would require 287 (255 + 32) bytes, and the maximum number of cursors possible using that index would be 114 (32 kilobytes divided by 287). That number would be reduced if the DB2 Server for VSE requests caused internal cursors to be created. Internal cursors are always smaller than 50 bytes, and cannot use index keys.

If you have many complex requests, you may have to increase NCSCANS. If it is not set to a high enough value, users will get SQLCODE -522. For information on the virtual storage used by NCSCANS, see the *DB2 Server for VM Program Directory* .

LTIMEOUT

This parameter specifies a general lock wait timeout period for any SQL application, and especially as the way to avoid global deadlocks for DUOW applications.

The range of the LTIMEOUT value is 0 to 99999 seconds. The value of zero indicates that no lock timeout should be enforced for agents connected to this database. This is the default value for a database.

A nonzero lock timeout value will cause any agents waiting for a lock to have their current transaction rolled back when the lock timeout period has expired. The agent will notify the application that a lock timeout has occurred with SQLCODE -911 (SQLSTATE 40001). A reason code will be returned to indicate whether it is a deadlock or lock timeout situation (reason code 2 for a deadlock situation and reason code 68 for a lock timeout situation). The lock timeout period begins at the moment an agent requests a lock on any database resource. The full lock timeout period is allowed for each lock request.

The lock timeout control parameter should be adjusted in those environments where lock contention between applications has started to affect the desired performance and concurrency levels.

If a lock timeout is required for your environment, it is recommended that your starting value be equivalent to the maximum period of time that you want an application to wait for a lock.

Note: The LTIMEOUT parameter is changed through the SET operator command. The timeout value will affect any users currently in LOCK WAIT. If a user has been in a LOCK WAIT for 100 seconds and the value of LTIMEOUT is set to a value less than 100, that user will receive a timeout. For more information on the SET operator command, see the *DB2 Server for VSE & VM Operation* manual.

If lock timeout control is activated, you should ensure that all applications recognize and can handle the -911 SQLCODE that may be received as the result of a lock timeout initiated rollback.

Note: New units of work that are waiting to begin because a log archive checkpoint is running or is scheduled to run are in a lock wait. The SHOW LOCK WANTLOCK operator command shows these units of work waiting to acquire an IX lock on the database. Because log archive checkpoints can potentially take a significant amount of time to complete, units of work in this particular type of lock wait are ignored by the lock timeout function.

PROCMXAB

This parameter specifies the number of times a stored procedure is allowed to terminate abnormally, after which a STOP PROC ACTION REJECT is performed against the procedure and all subsequent SQL CALL statements for that procedure are rejected. Note that a timeout that occurs while waiting for a stored procedure server to be assigned for an SQL CALL statement is not included in this count.

PROCMXAB must be an integer between 0 and 255. The default, 0, means that the first abend of a stored procedure causes SQL CALLs to that procedure to be rejected. For production systems, you should accept the default.

PTIMEOUT

This parameter specifies:

- The number of seconds before DB2 Server for VSE & VM ceases to wait for an SQL CALL to be assigned to a stored procedure server. If the PTIMEOUT interval expires, the SQL CALL statement fails.
- The number of seconds before DB2 Server for VSE & VM ceases to wait for the START PSERVER command to complete. If the PTIMEOUT interval expires, a message is displayed and the START PSERVER command terminates.

The default for PTIMEOUT is 180.

Recovery Parameters

LOGMODE

This parameter determines whether archives will be taken for the database and the log. Specify LOGMODE=A to maintain an archive of the database, LOGMODE=L to maintain an archive of the log, and LOGMODE=Y if you want logging but do not want the log archived.

LOGMODE=A allows you to restore the database and apply the current log. LOGMODE=L allows you to maintain a database archive as well as log archives. The database archive followed by the log archives are applied during restore, then the current log is applied.

Use LOGMODE=A or L if it is important to protect the database against media (DASD) failures; otherwise use LOGMODE=Y.

Note: Each sequence of log archives must be preceded by a database archive, so if you use LOGMODE=L, you must occasionally take a database archive. You do not need to switch to LOGMODE=A to do so.

For more information on LOGMODE, see “Choosing a Log Mode” on page 148.

CHKINTVL

This parameter determines how often a checkpoint is taken. A *checkpoint* is an internal operation in which data and status information is written to permanent (DASD) storage, and a summary status record is written to the log data set. A checkpoint causes two important events:

- Storage pool space is freed.

  As updates to data occur, duplicate copies of changed data pages are maintained. These copies (called *shadow pages*) are kept in the storage pools of the pages that were changed. A checkpoint frees the shadow pages, and thereby frees the storage pool space where they are kept.
- Log space may be freed.

  If LOGMODE=Y, a checkpoint typically frees log space by moving the logical beginning of the log forward to the beginning of the oldest LUW still active at the time of the checkpoint. If LOGMODE=A or L, log space is only freed when an archive is taken; not on every checkpoint.

Checkpoints are taken periodically: however, by the time one is taken, there may be a large amount of data to be committed. If a failure should occur before it is committed, much processing may need to be redone after the database is restored.

The CHKINTVL parameter lets you take checkpoints at predetermined intervals. Its value is specified in terms of the number of log pages written between checkpoints. You can use the SET operator command to change the value of CHKINTVL without having to stop and restart the application server. See the *DB2 Server for VSE & VM Operation* manual for more information on the SET operator command.

Note: Any changes you make using the SET command are only in effect while the application server is running. If you stop and restart the application server, it will use the settings you specified in the startup procedure.

By setting it low, you minimize the risk of filling the log or storage pools. However, because checkpoints are time-consuming operations that suspend SQL processing until they are completed, they should be taken infrequently. For more information on setting CHKINTVL, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

SLOGCUSH

This parameter defines the point at which the log cushion is entered and log-full processing begins. Its value is expressed in terms of the percentage of the log size. The default of 90 means that when the log is 90% full, log-full processing will be initiated.

In log-full processing, the oldest active LUWs are rolled back until enough log space is freed to bring the percentage of the log in use below the SLOGCUSH level. Ideally, checkpoints and archiving would continually free log space so that the log would never reach the SLOGCUSH level.

If the log should become 100% full, the database manager would end abnormally, so you should set SLOGCUSH to a value that allows log-full processing to take effect (free some log space) before this happens. If the database manager is ending with log-full conditions, you may want to lower the SLOGCUSH value or increase the size of your log data sets.

ARCHPCT

This parameter can be used to define a point at which an archive is automatically initiated or when an attempt to switch to the inactive log is automatically initiated. It is used only when LOGMODE=A or L is specified. Like SLOGCUSH, its value is expressed in terms of a percentage of the log.

Archives free up log space; however, they take some time to complete. If the SLOGCUSH value is reached during an online archive operation, all SQL processing is suspended until the archive is done. For this reason, it is best to ensure that archives are initiated in time to finish before the log fills to the SLOGCUSH percentage. This is done by setting the value of ARCHPCT lower than the value of SLOGCUSH.

When the log becomes full to the ARCHPCT value and alternate logging is not enabled, a message is issued to the VSE system operator to mount an archive tape and identify the *cuu* of the tape drive. The database manager then takes a database or log archive depending on whether you have LOGMODE set to A (database) or L (log).

If alternate logging is enabled, an attempt is made to switch to the inactive log. If the switch cannot be made because the inactive log has not been archived, the archive of both the inactive log and the active log will be initiated.

Normally, the operator explicitly archives the database or the log before the ARCHPCT value is reached, by issuing one of the archive commands. If the ARCHPCT is reached, meaning that the log is almost full, the action that the database manager takes depends on the LOGMODE that is in effect. See Table 6 for a summary of these actions.

Table 6. Summary of Activity When ARCHPCT Level Is Reached

LOGMODE Parameter	Activity When ARCHPCT is Reached
A	An operator message is issued that requests a database archive.
L and ALTLOG=N	An operator message is issued that requests a log archive.
L and ALTLOG=Y	An attempt is made to switch to the inactive log.
Y	Because the log cannot be archived, the value for ARCHPCT is ignored. When the log is full it wraps. If an LUW spans the entire log, a ROLLBACK WORK is forced for that LUW.

Note: To see how full the log is, you can issue the `SHOW LOG` command. For a description of this command, see the *DB2 Server for VSE & VM Operation* manual.

TAPEMGR

This parameter indicates whether there is a tape manager available to handle tape assigns during database and log archives. Y indicates there is a tape manager; N indicates there is no tape manager.

If TAPEMGR=N, the operator is prompted to enter the virtual device address for the database or log archive (with message ARI0299A).

If TAPEMGR=Y, the tape assign is handled by the tape manager and the operator is not prompted to enter the virtual device number (cuu) of the database or log archive.

ARCHTAPE

This parameter controls when archive tapes are unloaded. UNL indicates that at the end of writing to each tape of a log or database archive, the tape will be unloaded from the tape drive. This occurs regardless of whether the archives are dynamically or statically assigned; REW indicates that only archives taken using dynamic allocation will be unloaded at end of tape. If it is the last tape of the archive, and end of tape has not been reached, the tape will remain in the drive and must be unloaded manually by the operator.

SOSLEVEL

This parameter defines the storage cushion for storage pools. Its value is expressed as a percentage of space remaining in a storage pool. In multiple user mode processing (and single user mode processing where LOGMODE is not N), if any storage pool gets full to the point where only the SOSLEVEL percentage of storage pool pages is still free, a checkpoint is taken to free any shadow pages in use.

If, following this, only enough pages are freed to bring the number of free pages just above the SOSLEVEL, frequent checkpointing could occur. For more information, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual. If, however, the number of free storage pool pages is still at or below SOSLEVEL, message ARI0202I is issued once to inform the user that the number of free pages left in the storage pool is fewer than the SOSLEVEL. This message is also issued once in single user mode with LOGMODE=N, but no checkpoint is taken.

Attention: If message ARI0202I is received, it indicates some action may be needed to prevent imminent filling of the storage pool.

One possible action is to stop the application server and extend that storage pool by adding dbextents to it. However, you can remedy the situation without stopping the application server if you have set SOSLEVEL high enough to give you adequate warning. When the message is received, proceed to remove unneeded data from the storage pool, either by dropping dbspaces or tables, or by reorganizing the data with a smaller percentage of free space for each page. In order to do this, you must have adequate warning to schedule the necessary processing.

Service Parameters

DSPLYDEV

This parameter:

- Defines where certain informational and error messages are to be routed
- Governs the display of startup messages including initialization parameters derived from combining the DB2 Server for VSE defaults, parameters read from a source member, and parameters specified with the EXEC command or job control statement
- Controls the routing of any DB2 Server for VSE mini-dumps and shutdown messages.

Depending on the operating procedures at your installation, set DSPLYDEV=C if the output should be sent to SYSLOG; to L if the output should be sent to SYSLST; and to B if the output should be sent to both SYSLOG and SYSLST.

Generally, you will want to set DSPLYDEV to either C or B for debugging a problem from the operator's console; and to L for normal operations.

DUMPTYPE

This parameter defines whether or not dumps are to be taken, and the amount of information to be dumped if they are.

DUMPTYPE=N indicates that a dump is not taken.

DUMPTYPE=F gives you a full partition dump on some error conditions, including trace points.

DUMPTYPE=P gives you a partial dump of the database partition, excluding major phases (read-only code), on certain error conditions. A dump is *not* taken when a limit error (message ARI0039E) or hardware error (message ARI0041E) occurs, or when a user specification error is detected. It is recommended that you always use DUMPTYPE=F and not DUMPTYPE=P, as partial dumps generally do not contain enough information for debugging problems. The partial dump output is also generated to the trace tape for some trace points.

Generally, you should not run with DUMPTYPE=N (no dumps), but you may find occasions when you want to prevent dumping.

You can use the SET operator command to change the value of DUMPTYPE without having to stop and restart the application server. See the *DB2 Server for VSE & VM Operation* manual for more information on the SET operator command.

Note: Any changes you make using the SET command are only in effect while the application server is running. If you stop and restart the application server, it will use the settings you specified in the startup procedure.

For more information on dumps, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

EXTEND

This parameter specifies whether or not special recovery commands are processed at startup. Only set it to Y when you have a DBSS processing error or a severe user error. For more information on this parameter, see the discussion on starting the application server to recover from DBSS errors in the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

TRACDBSS, TRACRDS, TRACWUM, TRACDRRM, TRACDSC, TRACCONV, and TRACSTG

These parameters call the trace facilities during startup (as opposed to the TRACE operator command). Except for TRACWUM and TRACDRRM (which are not supported in single user mode), they are used primarily for tracing in single user mode, but can be set in multiple user mode if you want to start tracing as soon as possible. For information about tracing, refer to the *DB2 Server for VSE & VM Operation* manual.

Security Auditing: The TRACRDS parameter is also used to start security audit tracing during startup. A security audit is a special case of the normal trace facility. Unlike other traces, which are usually only started for problem determination, a security audit trace may be continually active while the database manager is running. This may be a standard procedure for some installations.

If you do not want the security audit to be continually active, you can start and stop it with the TRACE operator command instead.

For more information, see the discussion on security auditing in the *DB2 Server for VSE & VM Database Administration* manual.

TRACEBUF

This parameter specifies the amount of memory (in kilobytes) to allocate to the trace buffer. Specifying a nonzero value causes trace output to be stored in a fixed size buffer in memory. Trace records are stored in wrap-around mode in this buffer, and when tracing is turned off, the contents of the buffer are written to disk or to tape (as specified by the ARITRAC FILEDEF statement). The trace buffer is only created if you specify TRACEBUF with at least one of the startup initialization parameters TRACRDS, TRACDBSS, TRACDSC, TRACCONV, TRACDRRM, TRACWUM, or TRACSTG; it is not created if the TRACEBUF default (n=0) is specified. A suggested size for the trace buffer is 100 kilobytes or more. If you do not specify TRACEBUF and tracing is requested, trace records are written directly to disk or tape as the trace points are processed.

Single User Mode Initialization Parameters

Table 7 identifies the initialization parameters that apply when the database manager is operating in single user mode.

Table 7. Single User Mode Initialization Parameters

Parameter	Default	Minimum	Maximum
Environment Parameters			
DBNAME=name	From DBNAME directory	—	—
SYSMODE=S	S	—	—
STARTUP=W|R|F|U	W	—	—
PARMID=name	None	—	—
CHARNAME=name	INTERNATIONAL	—	—
ACCOUNT=T|D|E|N	N	—	—
DBPSWD=password	None	—	—
PROGNAME=name	None	—	—
DSPSTATS=nn	00	00	21
Performance Parameters			
NPACKAGE=n	10	1	32766
NPACKPCT=n	30	0	100
NPAGBUF=n	10 + NCUSERS x 4	10	3500
NDIRBUF=n	NPAGBUF	10	28000
NCSCANS=n	30	1	655
Recovery Parameters			
LOGMODE=Y|A|N|L	Y	—	—
CHKINTVL=n	10	1	99999999
SLOGCUSH=n	90	11	90
ARCHPCT=n	80	10	99
TAPEMGR=N|Y	N	—	—
ARCHTAPE=REW|UNL	REW	—	—

Table 7. Single User Mode Initialization Parameters (continued)

Parameter	Default	Minimum	Maximum
Environment Parameters			
SOSLEVEL=n	10	1	100
Service Parameters			
DSPLYDEV=L|C|B	L	—	—
DUMPTYPE=P|F|N	F	—	—
EXTEND=Y|N	N	—	—
TRACDBSS=nnn...	000...	000...	222...
TRACRDS=nnnnnnn	0000000	0000000	2222222
TRACDSC=nn	00	00	22
TRACCONV=n	0	0	2
TRACSTG=n	0	0	1
TRACEBUF=n	0	0	99999

Most of the considerations for setting these parameters are the same as those described under "Multiple User Mode Initialization Parameters" on page 47, with the following exceptions:

- The RMTUSERS parameter does not apply.
- The SYNCPNT parameter does not apply.
- The value of SYSMODE is S, which specifies that the database manager is dedicated to a single application.
- The database manager does not generate accounting records when STARTUP=C|E|L|S|I|M|P, which are special situations. For more information, see the *DB2 Server for VSE & VM Operation* manual.
- The PROGNAME parameter is required (except when STARTUP=C|E|L|S|I|M|P, which are special cases), to identify the application program to be run.
- The NCUSERS parameter is not used; it defaults to 1.
- The DISPBIAS parameter does not apply.
- The NLRBS and NLRBU parameters are omitted (there is no locking in single user mode).
- The LOGMODE parameter can take the value N, which specifies that changes made by the application program are not to be logged.

  If LOGMODE=N, database changes are only committed when a checkpoint is explicitly taken (with COMMIT WORK statements).

  The ARCHPCT parameter cannot be specified if LOGMODE=N.
- The TRACDRRM and TRACWUM parameters do not apply.

Tape Support

The database manager can write archive information, trace output, and accounting output to tape files. You can use tape files for input and output to the DBS utility.

To assign tape files, use the usual TLBL job control statements. Most installations place these statements for the trace, database archive, log archive, and accounting tapes in the catalogued procedure used to identify the database data sets. An example of this is under “Step 3: Setting Up Your Database Job Control” on page 217. Other examples are shown throughout the manual as needed.

Note: All tapes used by the database manager **must** use IBM standard (EBCDIC) labels. Unlabeled tapes are not supported.

Tape Manager Support

If you have a tape manager, you can take advantage of it when performing archives. The initialization parameter TAPEMGR indicates whether a tape manager is being used. During a database or log archive, if a tape manager exists and TAPEMGR = Y, the tape manager handles the tape assign, eliminating the need to enter the virtual device address of the archive output manually.

The archive initialization parameter ARCHTAPE enables you to specify whether the database should unload archive tapes upon completion of the archive. Also, if archives are statically assigned, each tape that is part of a multivolume database or log archive can be unloaded upon completion of writing to each tape. This parameter allows the operator to perform archives with minimal operator intervention.

Starting the Application Server in Multiple User Mode

When the application server is started in multiple user mode, operator commands can be issued and the operator may receive messages requesting that specific actions be done (for example, mounting a tape).

If you have a single database, start the application server like any batch job, by submitting job control statements or entering an EXEC command for ARISQLDS from the system operator console. To simplify the startup process, keep the DLBL and TLBL job control statements in the standard label area.

The job control example in Figure 17 shows how to start the application server by allowing the default initialization parameters to set up a normal multiple user mode environment.

```
// JOB MULTI
// EXEC PROC=ARIS75PL
// EXEC PROC=ARIS75DB
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='DBNAME=SQLDB1_NEWYORK_INV'
/*
/&
```

Figure 17. Job Control to Start in Multiple User Mode

Notes:

1. ARIS75PL is a cataloged procedure that contains the LIBDEF statements that refer to the DB2 Server for VSE production libraries. It is updated during the initial installation of the database manager. See the *DB2 Server for VSE Program Directory* manual for instructions on how to create this procedure.
2. ARIS75DB is a cataloged procedure that contains the DLBL job control for operating the database manager on your database. It (or one similar to it) is defined when you generate the database and should be updated when (or if) you add dbextents. For more information, see “Step 3: Setting Up Your Database Job Control” on page 217.
3. For procedures ARIS75PL and ARIS75DB, substitute your own procedures or job control statements to identify the database and the production libraries.

If you have generated more than one DB2 Server for VSE database, you would have a job control procedure for each (for example, ARIS75DB, DBNAME01, and DBNAME02). To start the application server, you would reference the appropriate cataloged procedure.

There is nothing special about using cataloged procedures as shown in Figure 17: this is just one of the ways you can use VSE job control facilities to run the database manager. For other techniques, see your VSE manuals.

The job control statements and procedures must include:

- An EXEC statement to run the program ARISQLDS. This statement can optionally specify parameters for overriding the default initialization parameters.
- A DLBL statement for the database directory, which must refer to the directory data set (BDISK) defined for the database being accessed.
- A DLBL statement for each log. Although one log is adequate, two are recommended to protect against media failures on a log. The logs are named LOGDSK1, LOGDSK2, ALTLGD1, and ALTLGD2.
- One DLBL statement for each dbextent currently defined for the database. The dbextents are named DDSK1, DDSK2, ..., DDSK*nnn*.
- LIBDEF (and, perhaps, DLBL and EXTENT) statements that identify the production libraries and any other needed libraries (for example, user applications for exits).
- If you intend to use archiving, a TLBL statement for the database archive file. The file name on this statement must be ARIARCH.
- If you intend to use log archiving, one TLBL statement for the log archive file for the active log data set, and one for the database archive file that must precede each sequence of log archives. If alternate logging is enabled, a TLBL statement will be needed for the log archive file for the inactive log data set. The file name on the TLBL statements must be ARILARC (active log data set) and ARILALT (inactive log data set).
- If you intend to use tracing, job control statements for the trace output file.

  The trace output file can be a tape file or a DASD file. If it is to be a tape file, you need a TLBL statement, and the file name on this statement must be ARITRAC. If it is to be a disk file, you need a DLBL, an EXTENT, and an ASSGN statement. If the disk file is managed by the VSE/VSAM Space Management for SAM Feature, the EXTENT statement is optional, and the ASSGN statement is not applicable. The file name on the DLBL statement must be ARITRAC. For examples of job control statements that you can use for tracing to disk, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.
- If you intend to use the accounting facility, a job control statement for each accounting file. (You can use just one accounting file, but two are recommended.) An accounting file, like a trace output file, can reside on tape or disk. For an example of job control statements that can be used with the accounting facility, see "Setting Up a Job Control for the Accounting Files" on page 187.

Running Multiple User Mode Application Programs

Batch Application Programs

When the application server is started in multiple user mode, batch SQL application programs can be started by normal means. Figure 18 on page 71 shows the job control statements for starting a batch program and for passing user parameters directly to that program.

Note: If you plan to run your application programs in both multiple user mode and single user mode, follow the protocols discussed in the section "CALL/RETURN Protocols for Application Programs in Single User Mode" on page 74.

```
// JOB USER PROGRAM WITH USER PARMS
// EXEC USERPROG,SIZE=AUTO,PARM='parm1,parm2'
/*
/&
```

Figure 18. Job Control to Start a Batch Application Program

Notes:

1. The AUTO keyword is not required, but is recommended.
2. The user program must be preprocessed by the database manager before being run.

VSE/ICCF Application Programs

Running SQL application programs under VSE/ICCF is the same as running any program except it is recommended that you set GETVIS to AUTO on the /OPTION statement. To run user SQL programs, the VSE/ICCF user would do an /EXEC for the file containing the VSE/ICCF control statements. These statements load the program and specify the GETVIS=AUTO option, as suggested in Figure 19.

```
/LOAD MYPROG,PARM='...'
/OPTION GETVIS=AUTO
/DATA
   •
   •
   •
```

Figure 19. Example of VSE/ICCF Control Statements for Running an Application Program

CICS Transactions

All CICS transaction programs written to use the database manager can be called using any of the means available under CICS. Usually, transactions are called directly from a terminal. If the terminal user has signed on to CICS, the CICS user ID is used as the default user ID for SQL operations done by the transaction only if the transaction does not issue an SQL CONNECT statement. If the terminal user has not signed on, the default user ID for CICS users is used. This ID was defined when online support was started through the CIRB or CIRA transaction. For a more complete discussion, see the *DB2 Server for VSE & VM Application Programming* manual.

CICS transactions can also be initiated by other means not directly tied to users. In these situations, the default user ID will be the one defined by the CIRB or CIRA transaction.

For CICS SQL transactions to be run, the database manager must be running in multiple user mode, and the online support must have been started with either the CIRB or CIRA transaction.

CICS Pseudo-Conversational Transaction Considerations: "Pseudo-conversational" refers to a technique for coding CICS transactions that interact with a user at a terminal. The transaction is not active while it is waiting for the user to enter a response. The following scenario shows one method of coding a pseudo-conversational transaction:

1. The transaction writes a question on the terminal.
2. The transaction issues a CICS RETURN with the TRANSID parameter specified.
3. The user enters the response.
4. CICS restarts the transaction automatically.
5. The transaction interprets the response.
6. The transaction can then ask another question or end.

ISQL is not pseudo-conversational. It runs as two CICS transactions (named ISQL and CISQ). When a long-running SQL statement is being processed, the ISQL transaction:

1. Issues message ARI7044I:
   ```
   Command in progress. Terminal is now free.
   ```
2. Times out
3. Ends with a CICS RETURN without the TRANSID parameter.

This allows the user to enter ISQL CANCEL to cancel a long-running SQL statement. If this is not done, the ISQL transaction is restarted by the CISQ transaction when the SQL statement completes. The results of the SQL statement are displayed when the ISQL transaction is restarted by the CISQ transaction.

If the ISQL transaction times-out (ends with message ARI7044I) and a pseudo-conversational transaction is started, the following events can cause confusion:

- If the pseudo-conversational transaction requests input from the user (and issues a RETURN with TRANSID specified), the user is not able to cancel ISQL with an ISQL CANCEL command, because the ISQL CANCEL is interpreted by CICS as data to the pseudo-conversational transaction.
- If the pseudo-conversational transaction requests input from the user (and issues a RETURN with TRANSID specified), and processing of the long-running SQL statement ends before the user completes input and presses the ENTER key, ISQL will display the SQL statement output on the terminal. This will overlay any input that the user may have typed on the terminal for the pseudo-conversational transaction. The transaction is waiting for input, but the request for the input has been overlaid by the ISQL output. The pseudo-conversational transaction continues to wait for input, so the user can enter input to it after the ISQL transaction either times-out or ends.

To eliminate this confusion, avoid running pseudo-conversational transactions and ISQL on the same terminal at the same time.

Starting the Application Server in Single User Mode

An application program running in single user mode runs in the database partition under the control of the database manager. The application server is started in single user mode (SYSMODE=S), and the program name is provided as an initialization parameter (PROGNAME=*name*).

Figure 20 shows an example. When the application server is started, it passes control to the application program specified by the PROGNAME parameter. All other initialization parameters are allowed to default.

Note: The PROGNAME parameter is not used if STARTUP is specified as C (database generation), E (adding dbextents), L (log reformatting or reconfiguration), S (adding dbspaces), I (reorganization of catalog indexes), or M (catalog migration). These types of startup specify the operation to be performed, so a program name is not needed.

```
// JOB SINGLE
// EXEC PROC=ARIS75PL
// EXEC PROC=ARIS75DB
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,PROGNAME=name'
/*
/&
```

Figure 20. Job Control to Start in Single User Mode

Note: It may be necessary to specify SIZE=(AUTO,*n*K) for ARISQLDS, where the *n*K value specifies the amount of storage required by the programming language to load run-time routines and perform dynamic storage allocation.

Specifying User Parameters

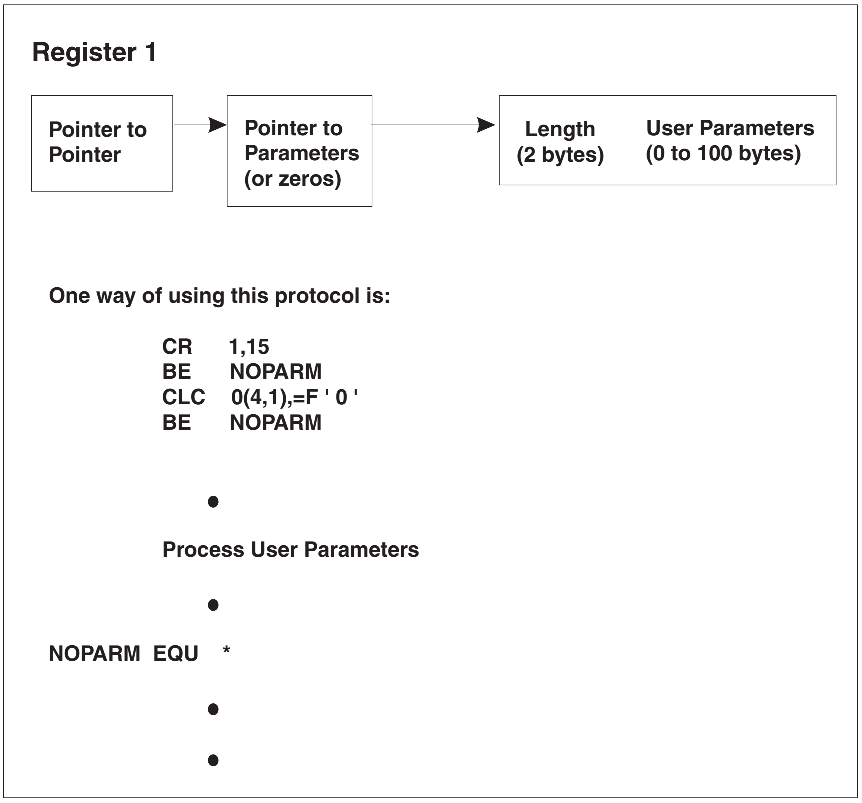
When starting the application server in single user mode, you can also specify user parameters to be passed to the application program, along with initialization parameters passed to the application server. A slash (/) must be placed between the application server parameters and the application program parameters, as shown in Figure 21.

```
// JOB SINGLE WITH USER PARMS
// EXEC PROC=ARIS75PL
// EXEC PROC=ARIS75DB
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,PROGNAME=PROG1/parm1,parm2'
/*
/&
```

Figure 21. Job Control to Start in Single User Mode and Provide User Parameters

Note: Up to 100 characters can be specified as parameters on the EXEC statement or command. Each parameter must be separated by at least one comma or blank; these commas and blanks also count as characters. The parameter string cannot extend past column 71. If a continuation is needed, column 72 must contain a continuation character (any non-blank character), and the parameters must continue in column 16 of the next line.

Parameters are passed to the application program by the standard VSE/ESA protocol, as shown in Figure 22. When the database manager has processed its initialization parameters, it passes the address (in register 1) of a pointer to the application program specified by the PROGNAME parameter. (If there are no user parameters, the pointer contains binary zeros.)


Register 1
Pointer to Pointer
Pointer to Parameters (or zeros)
Length (2 bytes)
User Parameters (0 to 100 bytes)
One way of using this protocol is:

```
         CR     1,15
         BE     NOPARM
         CLC    0(4,1),=F ' 0 '
         BE     NOPARM

              •

         Process User Parameters

              •

NOPARM  EQU   *

              •

              •
```

Figure 22. Passing User Parameters to a User Application Program

Application program parameters are not displayed along with the initialization parameters. Application program parameters also cannot be specified in a DB2 Server for VSE parameter data set.

CALL/RETURN Protocols for Application Programs in Single User Mode

In single user mode, an application is called using normal CALL/RETURN protocols, as follows:

Register 1
: Pointer to pointer to user parameters

Register 13
: Pointer to DB2 Server for VSE save area

Register 14
: Return point to the database manager

Register 15
: Entry point of the user program.

Note: This same protocol can also be used by programs running in multiple user mode.

Upon entry, the application program must store the registers in the DB2 Server for VSE save area, and restore them before returning control to the database manager. Failure to do so causes unpredictable results.

The database manager sets an abnormal task termination routine to intercept abnormal end conditions, including program checks. If the user program issues an STXIT AB macro instruction, this macro overrides the DB2 Server for VSE STXIT AB macro instruction. If the user program issues an STXIT PC macro instruction, this macro overrides the DB2 Server for VSE STXIT AB macro instruction for program checks. A user program should not issue one of these STXIT macro instructions.

Notes:

1. A PL/I application program can issue an STXIT PC macro instruction to override the DB2 Server for VSE exit. If this is done, the database manager cannot handle a program check, but it can still handle other abnormal end conditions.
2. When a user runs an application with the TRAP(ON) run-time option of LE/VSE and the DB2 Server for VSE application is running in single user mode, LE/VSE and DB2 Server for VSE keep track of calls to and returns from the database. If a program interrupt or abend occurs when the application is running, the LE/VSE condition manager is informed whether the problem occurred in the application or in the database manager. If the program interrupt or abend occurs in the database manager, the LE/VSE condition handler passes the condition back to DB2 Server for VSE. For more information, see the section “Condition Handling with LE/VSE” in the *DB2 Server for VSE & VM Application Programming* manual.

An application program should always return control to the database manager (if possible). It should never issue a CANCEL, DETACH, DUMP, JDUMP, RETURN or EOJ (or equivalent macro) instruction. These instructions prevent the database manager from doing its normal shutdown processing, such as closing the database files and the trace tape file (if one is being used).

The DB2 Server for VSE abnormal end routines issue CLOSE macro instructions for the database, the trace and accounting files if those facilities were activated, and the SYSLST file if it was opened. This same close process is also done when the application program returns control to the database manager at the end of the job. If you do not return control to the database manager, the files cannot be closed. However, the database manager does not have to close these files. The VSAM automatic close function will be started and the application server will still be accessible. When the application server is next started, VSAM may issue an informational message stating that the files were not closed in the previous run. If tracing or accounting were active, their output files may not have had the last buffer (or buffers) written. And, if the output files were on tape, no tape mark would be written.

The database manager uses an “eye-catcher” technique for determining when a specific module is in error. The eye-catcher is displayed in the DB2 Server for VSE mini-dump. An application program can use the same technique in single user mode, assuming that the DB2 Server for VSE abnormal end exit has not been overridden by a user STXIT AB or STXIT PC macro instruction. A suggested coding example in assembler language is shown in Figure 23.

```
        USING  *,15
        B      SKIPEYE        BRANCH AROUND EYE-CATCHER
        DC     AL1(16)        LENGTH OF CHARACTER STRING
        DC     CL8'progname'  PROGRAM NAME EYE-CATCHER
        DC     CL8'&SYSDATE'  DATE PROGRAM COMPILED
        DS     0H
SKIPEYE EQU    *
        STM    14,12,12(13)   SAVE  DB2 Server for VSE REGISTERS
        BALR   12,0           ESTABLISH BASE REGISTERS
        DROP   15
        USING  *,12
        LA     11,MYSAVEAR    GET ADDRESSABILITY TO MY SAVE AREA
        ST     11,8(13)       SAVE ADDRESS OF SAVE AREA IN  DB2 Server for VSE SAVE AREA
        ST     13,MYSAVEAR+4  SAVE ADDRESS OF  DB2 Server for VSE SAVE AREA IN SAVE AREA
        LR     13,11          SET REGISTER 13 TO MY SAVE AREA
            •
    Body of the Application Program
            •
EXIT    L      15,RETCOD      SET RETURN CODE (OR SET TO ZERO)
        L      13,4(13)       GET  DB2 Server for VSE SAVE AREA
        L      14,12(13)      GET  DB2 Server for VSE REGISTER 14
        LM     0,12,20(13)    GET OTHER  DB2 Server for VSE REGISTERS
        BR     14             RETURN TO DATABASE MANAGER
```

Figure 23. Use of an Eye-catcher by an Application Program

Notes:

1. The instruction BALR 15,0 can be used just ahead of the USING *,15 instruction as long as other registers are not used until the DB2 Server for VSE registers have been saved.
2. The techniques shown here work whether the application program is called by the database manager, or is called as a job itself. Thus, the same application program can be run in either single or multiple user mode.
3. The techniques shown here may not always be achievable by a FORTRAN, C, COBOL, or PL/I program. A program written in one of these languages may need to be called by a pre-entry routine, to ensure that register 15 contains a zero (or valid return code) upon return to the database manager. A PL/I program can use PLIRETC/PLIRETV.

Overriding Initialization Parameters

When starting the application server, you can change the default parameter values in either of two ways:

- You can specify the parameters in the PARM field of the job control EXEC statement or command
- You can create a DB2 Server for VSE parameter data set as an A-type source member, and invoke it with the PARMID initialization parameter. See Figure 15 on page 50 for an example.

You can also combine the two methods. Parameters specified in the parameter data set override the default values, while those specified with the EXEC override both the default values and those specified in the parameter data set. Thus, a user who has a parameter data set with an incorrect parameter value can override the error with a correct specification on the EXEC statement or command.

When all the values of the initialization parameters have been resolved, the final values (or defaults, if no values have been overridden) are displayed on SYSLOG, SYSLST, or both (according to the value of the DSPLYDEV parameter).

You can use up to 100 characters on the EXEC command for specifying parameters. Separate each parameter by at least one comma or blank, but the commas and blanks also count as characters. An example of specifying parameters with the EXEC command is:

```
EXEC ARISQLDS,SIZE=AUTO,PARM='DSPLYDEV=B,DUMPTYPE=F'
```

You can use up to three 'PARM=' clauses of the EXEC statement for specifying parameters. Each 'PARM=' can be up to 100 characters. If PARM='value' was specified twice or three times, the values are concatenated according to their sequence.

As an option, you may choose to set up your initialization parameters in one or more A-type source members. Such an arrangement allows you to specify more user parameters (if any) when running application programs in single user mode. User parameters (those for the application program itself), cannot be specified in a source member, and must be specified in the PARM field of the job control EXEC statement or command. If you plan to use application program parameters, refer to "Specifying User Parameters" on page 73.

Creating a Parameter Data Set

You can store various parameters in A-type source members. You can have as many A-type source members as you need. Each one can start the application server for a slightly different environment. To use the parameters, specify the member name in the PARMID initialization parameter. Figure 24 shows an example of a job that catalogs a source member.

```
// JOB CATALPRM
// EXEC LIBR
   ACCESS SUBLIB=LIBRARY.SUBLIB
   CATALOG PARMXMPL.A
   DBNAME=SQLDB1_NEWYORK_INV,RMTUSERS=50,
   DSPLYDEV=B,NDIRBUF=20,SYSMODE=S,
   PROGNAME=USERPROG,NPAGBUF=20,
   DUMPTYPE=F                  COMMENT - FULL PARTITION DUMP
   NCSCANS=20
/+
/*
/&
```

Figure 24. Job to Catalog a Source Member

The rules for specifying parameters in a member are a little different from those specifying parameters in the job control. In particular:

- The parameters must be in uppercase in a parameter file.
- A blank after a parameter ends the processing of the line, so do not put a blank between parameters -- anything on the line after the first blank will be ignored. You can, however, use blanks to put comments in the member, as shown for the DUMPTYPE parameter in Figure 24.
- A comma at the end of a line is not required, but can make the statement easier to read.
- User parameters (those destined for the application program itself), are not allowed in a member containing DB2 Server for VSE initialization parameters. If the database manager detects any parameters other than its own initialization parameters, it issues error messages and stops.

Stopping the Application Server

This section discusses the following topics:

- Taking an archive
- Verifying the directory
- Online support considerations

In single user mode, the application server stops itself when the task is completed. In multiple user mode, the operator stops it by issuing the SQLEND operator command. In both modes, the database files and the trace file (if active) are closed. The SQLEND command is described in the *DB2 Server for VSE & VM Operation* manual.

The SQLEND command can be entered from the VSE system operator console only. Its format is shown in Figure 25. The ARCHIVE, LARCHIVE, and UARCHIVE parameters are used to initiate archive activities after the database has been shut down, and are discussed in the next section. The NORMAL parameter is used to shut down the database when all work in progress is completed. The QUICK parameter is used to stop all work in progress and shut down immediately. The TRCPURGE parameter is used if you want to purge the contents of the trace buffer at DB2 Server for VSE shut down. You can also specify the DVERIFY parameter to do a directory verification.

```
           ┌─NORMAL───┐
►►──SQLEND─┼──────────┼──┬─────────┬──┬──────────┬──────────────────────────────────►◄
           ├─ARCHIVE──┤  └─DVERIFY─┘  └─TRCPURGE─┘
           ├─LARCHIVE─┤
           ├─UARCHIVE─┤
           └─QUICK────┘
```

Figure 25. SQLEND Operator Command

Taking an Archive

The SQLEND command can be set up to enable the operator to take a database or log archive after all DB2 Server for VSE activity has stopped. The following parameters are available for archiving:

- ARCHIVE for a database archive using DB2 Server for VSE facilities
- LARCHIVE for a log archive using DB2 Server for VSE facilities
- UARCHIVE for a database archive using user facilities.

Attention: User archive facilities are available for the database, but not the log. Never attempt to use user facilities to archive a log.

The most appropriate time to take an archive is at shutdown, so consider setting up a procedure for periodic SQLENDs with the ARCHIVE, UARCHIVE, or LARCHIVE parameters, as needed.

For both database and log archives, online archives are disruptive to users. Taking archives during SQLEND avoids this disruption. In addition, database archives taken at SQLEND contain data that is consistent, whereas those started by operator ARCHIVE commands or triggered by ARCHPCT typically contain uncommitted or incomplete data, and require information from the log to make the data consistent. (Consistency is not a problem for log archives regardless of when they are taken, because the database manager always waits until all LUWs end before taking the checkpoint on which the log archive is based.)

To determine the best recovery procedures for your installation, see “Recovering from DASD Failures that Damage the Database” on page 147.

If the operator specifies ARCHIVE or UARCHIVE when LOGMODE=Y, the database manager automatically switches the LOGMODE to A. To resume running with LOGMODE=Y, the operator must do a COLDLOG. See “Switching Log Modes” on page 167.

Should you decide *not* to take an archive at shutdown, specify NORMAL or QUICK. During a normal shutdown, the database manager allows all active LUWs to finish before ending. During a quick shutdown, the application server ends immediately: in-progress LUWs receive a negative SQLCODE and are rolled back the next time the application server is started.

Note: A User Archive will **NOT** be consistent if it is taken following an SQLEND QUICK shutdown.

If you are running with LOGMODE=L, and request a database archive, and if there is data in the log, then the database manager takes a log archive before taking the database archive. If alternate logging is enabled, a check will be done to see if the inactive log was previously archived. If it was not, it will be archived before the active log. The log archives are written to tape.

Database archives are written to tape. When running a database archive, the database manager displays external label information for you to write on the tape. It then requests that you mount the required tape volumes. See “Archiving Procedures” on page 152 for more information.

When the SQLEND command is issued with the NORMAL, ARCHIVE, LARCHIVE, or UARCHIVE parameters, a shutdown is not initiated until all users are disconnected from the application server. The database manager displays a message showing how many agents are still active. (An agent is an internal representation for a user.) As each agent becomes inactive, another message is displayed with an updated count.

The initial count displayed in the message includes all *active* user agents. When users who are inactive (not allocated to a real agent) disconnect from the database manager, no message is displayed to indicate a reduction in agents; the message is issued only when a user disconnects from the database manager while still allocated a real agent. This results in gaps in the updated count messages.

After issuing an SQLEND command, and before shutdown commences, the operator can issue a `SHOW ACTIVE` command to find out who is still using the database manager. Users who are connected with no active LUW can prevent the database manager from performing shutdown operations. For example, an ISQL user can end an LUW and then leave the terminal without exiting from ISQL. To determine whether inactive users are preventing the shutdown operation, use the `SHOW USERS` operator command to determine which users are still active. For more information on the SHOW commands, see the *DB2 Server for VSE & VM Operation* manual.

If the SQLEND command is issued with the QUICK parameter, all in-progress work ends and return code 508 is displayed on the console. This command can be issued at any time, even following an SQLEND issued with another parameter.

Verifying the Directory

The DVERIFY parameter determines whether the database manager checks for inconsistencies in the directory. It can be specified with the other parameters, but is ignored if you specify QUICK. It should be specified each time the database is archived (using either DB2 Server for VSE or user facilities); if it is not, any inconsistency in the directory will be recorded in the database archive, so a subsequent restore operation using that archive would fail.

Even if you have not requested a database archive, you should periodically verify the directory (perhaps every few days, depending on the volume of update activity). Otherwise, inconsistencies may surface later. For example, an inconsistency can cause an abnormal end during checkpoint processing. Early detection reduces data loss.

If an error is found in the directory, a message is displayed. If this happens, and you had specified ARCHIVE, the archive is not taken. If you had specified UARCHIVE (a database archive using user facilities), then when you are prompted to take the archive, do not do so. However, if you had specified LARCHIVE, the log archive *is* taken; the inconsistency in the directory does not affect the log, so the log archive is still valid. For information on recovering from directory verification errors, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

Online Support Considerations

If you are supporting an online (CICS) environment, you should stop the online support before ending the application server, in order to clean up CICS transaction processing efficiently. To stop the online support, enter the CIRR or CIRT transaction. For more information on the effect of a shutdown on online applications, see "Stopping the Online Support -- The CIRT Transaction" on page 112 and "Removing Connections -- The CIRR Transaction" on page 100.

Note: For DB2 Server for VSE, each link from the Online Support requires a dedicated agent, whether or not these agents are actually active. SQLEND NORMAL will not terminate these connections.

Chapter 5. Operating the Online Support

This chapter explains how to enable VSE guests to access an application server on a VM operating system, and how to operate the VSE online support.

Operating VSE Guest Sharing

Your VSE online users can access an application server on a VM host operating system when the VSE operating system is running as a guest in a virtual machine. (DRDA support is not provided with the VSE guest sharing function.) Database switching is supported for CICS online applications, which means that one resource adapter in one CICS region can connect to multiple application servers. Any CICS transaction in the CICS region can connect to any of the DB2 Server for VM application servers to which the online resource adapter has established connections. This means that:

1. Different transactions in a CICS region will be able to connect to different DB2 Server for VM application servers
2. Single transactions will be able to connect to different DB2 Server for VM application servers in different units of work.

The DB2 Server for VM application server can be accessed by specifying the server_name parameter on the CIRB transaction or on the CIRA transaction. The DB2 Server for VM application server must be defined in the DBNAME Directory. The DBNAME Directory provides the mapping of mapped DBNAME to *resid*. See "Choosing an Application Server Name" on page 23 *DB2 Server for VSE System Administration* for more DBNAME Directory information. The *resid* is the basic DBNAME, and must be the same as the one specified in the SET APPCVM command during the VSE initial program load. If there are multiple DB2 Server for VM servers on the VM host, there can be more than one SET APPCVM command.

The VM application server being accessed can be either on the same processor or on another processor in the network. For batch applications and for online users who want to access an application server on another processor in a SNA network, you must issue the SET APPCVM command when you start VSE. The command provides routing information for both batch and online users. Note that SET APPCVM is required only if VTAM is to be used in the connection. If the server and requester are in a TSAF collection on the same node, it is not necessary to issue the SET APPCVM command.

Figure 26 on page 82 shows the syntax of the SET APPCVM command.

```
►►──SET APPCVM TARGET──┬─resid────────────────────┬──────────────────────────────────────►◄
                       └─┤ avs_parameter_block ├──┘
```

avs_parameter_block:

```
├──(──resid,gateway_name,target_LU_name,mode_name──)─────────────────────────────────────┤
```

Figure 26. SET APPCVM Command

The variables have the following meanings:

resid
The resource identifier of the DB2 Server for VM application server which is the same as the *resid* parameter on the IUCV *IDENT entry in the database machine directory for VM operating systems.

avs_parameter_block
Only specify these parameters if the application server you want to access is in an SNA network. The names are defined by VTAM* statements when the network is built, and have these meanings:

resid
The resource identifier of the DB2 Server for VM application server. This is the same as the *resid* parameter on the IUCV *IDENT entry in the database machine directory on VM.

gateway_name
This corresponds to an APPL statement at the local system. To the SNA network, *gateway_name* is an LU with the same name.

target_LU_name
This corresponds to an APPL statement at the remote system.

mode_name
This corresponds to a mode table entry at the local and remote systems.

The parameters **must** be specified in the order shown above.

For more information about the AVS parameters, see the *VM/ESA: Connectivity Planning, Administration, and Operation* manual. For more information on the IPL SET APPCVM command in VSE, see *VSE/ESA System Control Statements*.

Note: The VSE Guest sharing facility requires 40KB of real storage for each database communication link. For more information on providing real storage, see *VSE/ESA System Control Statements*

Operator Responsibilities

VSE guest sharing is monitored from the VM console. All DB2 Server for VM operator commands can be used. In addition, in-doubt LUWs can be forced from the VM console.

Online support is required for ISQL and CICS transaction programs that access the application server. The DB2 Server for VSE online resource adapter must be started so that the application server can be accessed from the CICS online environment. If this is not done, and a CICS transaction attempts to access the application server, CICS will end the transaction with CICS/VSE abend code AEY9.

Operation of the online support involves the following:

1. Starting the application server in multiple user mode, either before or after CICS is started. (The online environment is not supported in single user mode.)
2. Starting the DB2 Server for VSE online support by running the CIRB transaction under CICS. The CIRB transaction accepts a list of server names. This allows online access to multiple application servers to be established from one command. CIRB enables the online resource adapter and also loads the online resource adapter above the 16MB line thus allows the memory space below 16MB to be used for other purposes. After CIRB has successfully completed its processing, the online resource adapter is ready to handle SQL requests from CICS transaction programs (such as ISQL).
3. After the online resource adapter is started, the CICS transaction CIRA can be used to add connections or enable online access to other application servers. CIRA can be entered multiple times with different *server_name*s. This establishes the connections or enables online access to the specified application server. CIRA also accepts a list of *server_name*s so that online access to multiple servers can be established with one command.
4. The transaction CIRR can be used to remove connections or disable online access to a particular application server or list of application servers. The online resource adapter is terminated if the CIRR transaction removes the connection or disables online access to the last application server.
5. Displaying information about active CICS transactions (including ISQL) that access an application server by using the CIRD transaction. The CIRD transaction accepts a *server_name* parameter to display the transactions accessing a particular application server. The * keyword can be specified to display all transactions on all of the application servers (for example, CIRD *).
6. Changing the default application server using the CICS transaction CIRC.
7. Stopping the online support without stopping either CICS or the application server by issuing the CIRT transaction. The CIRT transaction terminates all connections or access to all application servers and then terminates the online resource adapter.

If a local application server becomes unavailable for some reason, only the connections to that application server are lost. The online resource adapter remains active and connections or online access to other application servers can still be used. When the local application server becomes available again, the CIRA transaction can be used to re-establish connections to it. If there are any in-doubt LUWs associated with this application server, they will be resolved at this time.

If the default application server becomes unavailable, a new default server is not established automatically. Users attempting to connect to the default server will receive a message indicating that the server is not available.

These steps are described in detail below. For more information on starting and stopping online support for VSE guest sharing, see the *DB2 Server for VSE & VM Operation* manual.

Starting the Online Resource Adapter -- The CIRB Transaction

To activate the online support, run the CIRB transaction. When it completes, the resource adapter is enabled. Only when this happens can user transactions be executed.

CIRB has six parameters:

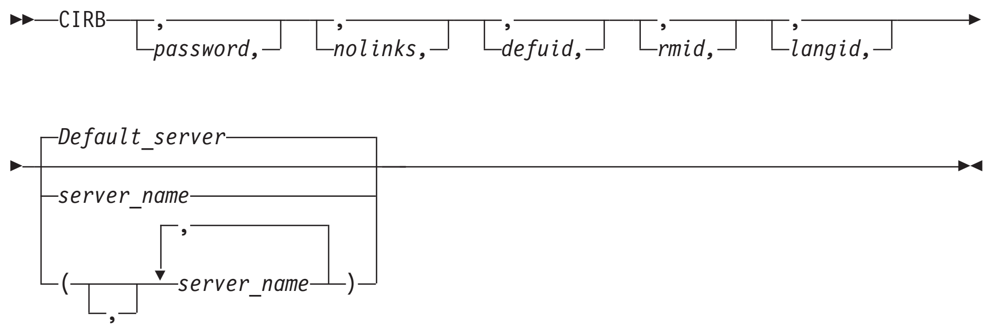


Figure 27. CIRB Transaction Syntax

The parameters are described in the following table:

Table 8. CIRB Transaction Parameters

Parameter	Default	Description

PASSWORD (positional parameter 1)	SQLDBAPW	This parameter establishes the operator's authority to activate online access to a local application server. The password identifies the CICS subsystem. The user ID of the subsystem is the CICS APPLID, which defaults to DBDCCICS. The procedure ARIS080D uses the following job control to give the password and user ID to the local application server:
		`// EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,`
		`LOGMODE=N,PROGNAME=ARIDBS'`
		`CONNECT SQLDBA IDENTIFIED BY SQLDBAPW;`
		`GRANT SCHEDULE TO DBDCCICS IDENTIFIED BY CICSPSWD;`
		`COMMIT WORK;`
		
		The password chosen (CICSPSWD above) must satisfy DB2 Server for VSE & VM specifications for a password. This password establishes which password to use when dropping connections through the CIRR or CIRT commands. See "Password Implications on Online Resource Adapter Termination" on page 116 for more details.

NOLINKS (positional parameter 2)	3	This parameter establishes the number of links (paths) that should be initialized to a local application server. Specify this parameter as a decimal value between 1 and 64. The number must be less than or equal to the value assigned to the NCUSERS initialization parameter of the DB2 Server for VSE & VM system. (The NCUSERS default is 5.)
DEFUID (positional parameter 3)	CICSUSER	This parameter identifies the default user ID used by the online support when it makes an implicit CONNECT to a local application server. This parameter must satisfy DB2 Server for VSE & VM specifications for a user ID.

Table 8. CIRB Transaction Parameters (continued)

Parameter	Default	Description
RMID (positional parameter 4)	0	This parameter identifies a unique resource adapter. You must specify it only if your installation has multiple CICS partitions active in the same VSE/ESA system, and if each CICS partition allows online access to a server. For the case of a local application server, recovery requires that the local server know the resource adapter it is servicing. You must specify this parameter as a decimal value between 0 and 63. If the DB2 Server for VSE online support detects that this ID is not unique in the system, it issues a message. The CIRB transaction then ends without enabling the resource adapter. There can be only one DB2 Server for VSE resource adapter enabled in a single CICS partition. An attempt to enable a second DB2 Server for VSE resource adapter causes the DB2 Server for VSE online support to issue a message, and the CIRB transaction ends without enabling the second resource adapter. The first one, however, remains in effect.
LANGID (positional parameter 5)	specified at installation	This parameter defines the language the DB2 Server for VSE online support uses to display error and information messages. The language you specify on this transaction becomes the default language for ISQL, CBND, C2BD, DSQG, DSQU, DSQD, and DSQQ. The ISQL welcome logo always appears in the language specified on this transaction. This parameter must take the form of a minimum 1-character, maximum 5-character language ID. You must use one of the language IDs in the LANGID column of the SQLDBA.SYSLANGUAGE table. The language ID must identify a language you have installed on the DB2 Server for VSE server. To choose another language, use the SET LANGUAGE command in ISQL. The following IDs can be specified on the CIRB transaction: **AMENG** American English **UCENG** Uppercase English **FRANC** French **GER** German **KANJI** Kanji (Japanese) **HANZI** Simplified Chinese If this parameter is omitted, the language defaults to the language chosen as the default at installation.
SERVER-NAME (positional parameter 6)	Determined from DBNAME directory or “SQLDS”.	This parameter enables you to specify the application servers that you want to access. If the list format specifies multiple servers, the first one in the list becomes the default server. Only the first server_name in the list may be omitted. If this parameter (or the first one in the list) is omitted, the default server is determined from the DBNAME directory. If the DBNAME directory does not specify a default server, then **SQLDS** becomes the default server name.

The CIRB transaction establishes the default application server. If the server_name parameter is not specified on the CIRB transaction, then the default server is determined from the DBNAME directory. If a single server_name is specified on the CIRB transaction then it becomes the default server. If a server_name list is specified on the CIRB transaction, the first server_name in the list becomes the default server. If the first server_name in the server_name list is blank then the default server is determined in the same way as when the server_name is omitted from the CIRB transaction. For example:

```
CIRB ,,,,,(,SQLMACH2)
```

This starts connections to two servers. The first one is the default server and its name is determined from the DBNAME directory or if it is not specified in the DBNAME directory it defaults to **SQLDS**. The second server is SQLMACH2.

Note that the following examples are not allowed. Only the first server_name in the list can be blank.

```
CIRB ,,,,,(SQLMACH2,)
CIRB ,,,,,(SQLMACH2,,SQLVM)
```

The number of server_names that can be specified on the CIRB command is limited by the size of the input line on the VSE console or a CICS terminal. The VSE console only allows one line of input. A CICS terminal allows much more input. If short server_names are used more can fit on the command. Server-names can be up to 18 characters long. If all of the required server_names cannot fit on the command, the CIRA transaction must be used to establish connections for the remaining server_names.

Figure 28 shows an example of using the server_name list on the CIRB transaction and from the message

```
"ARI0450I DB2 Server for VSE online support has an entry point of 02C02000"
```

we can find that the online resource adapter is loaded above the 16MB memory line.

```
msg f2
AR 015 1I40I READY
2 cirb ,,,,,(sqlmach1,sqlmach1)
F2-002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2-002 ARI0450I DB2 Server for VSE online support has an
                entry point of 02C02000 RMGL at 00541200.
F2-002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0055B2E0.
F2-002 ARI0458I The default server is SQLMACH1.
F2-002 ARI0457W Connections to SQLMACH1 already exist.
F2-002 ARI0402E Connections to SQLMACH1 could not be established.
```

Figure 28. Example of CIRB with Duplicate Server Names

The maximum number of application servers to which an online resource adapter can establish connections or enable online access to is only limited by the amount of storage available in the partition where the online resource adapter is running.

If you try to establish connections to an application server to which connections already exist, or to which online access is already enabled, the message *"ARI0457W Connections to <server_name> already exist."* is displayed. No action is taken against that server. If the connections to a local server need to be changed they must first be removed using CIRR or CIRT and then re-established using CIRA or CIRB. An example is shown in Figure 29 on page 87.

```
msg f2
AR 015 1I40I READY
2 cirb ,,,,,(sqlmach1,sqlmach2)
F2-002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2-002 ARI0450I DB2 Server for VSE online support has an
                entry point of 02C02000 RMGL at 00541200.
F2-002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0055B2E0.
F2-002 ARI0458I The default server is SQLMACH1.
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055C2E0.
2 cirr ,,,sqlmach2
F2-002 ARI0455I Connections to SQLMACH2 are disabled.
2 cira ,5,,sqlmach2
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055A2E0.
```

Figure 29. Example of Changing Connection Settings

Note that each local server in the list has its connections established with the same values for password, number of links, RMID, default user ID and language ID that were specified.

If the CIRB parameters for each server are identical, all of the connections or online access can be established with one CIRB transaction, as illustrated in Figure 30.

```
msg f2
AR 015 1I40I READY
2 cirb ,,,,,(sqlmach1,sqlmach2,sqlvm)
F2-002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2-002 ARI0450I DB2 Server for VSE online support has an
                entry point of 02C02000 RMGL at 00541200.
F2-002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0055A2E0.
F2-002 ARI0458I The default server is SQLMACH1.
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055C2E0.
F2-002 ARI0454I Connections to SQLVM established.
                RMCV at 0055D2E0.
```

Figure 30. Example of CIRB with Server-Name List

All three local application servers have the same number of connections, the same default user ID, the same password, the same RMID and the same language ID.

If one or more of the parameters must be different, then all of the connections cannot be established with one CIRB transaction. You will need the CIRA transaction to add additional servers.

If you enter a remote server name, which has been coded with *CONNPOOL=N* in the DBNAME directory, in the *server_name* parameter of the CIRB or CIRA transaction, CIRB or CIRA will not establish any links or sessions to the remote system where the remote server runs. The following message will not be displayed by CIRB or CIRA when it is processing such a remote server. The message will display for local servers and for remote servers with *CONNPOOL=Y*.

```
ARI0454I Connections to server_name established.
         RMCV at XXXXXXXX.
```

CIRB or CIRA will display the following message instead for every remote server processed at initialization time:

```
ARI0467I RMCV for remote server_name established.
         RMCV at XXXXXXXX.
```

Starting the CIRB Transaction

The CICS sequential device support can be used to automatically start the CIRB transaction when CICS is started. Either a CRLP (a card reader or line printer) device, or a sequential DASD device must be defined in the CICS DFHTCT, to allow them to simulate terminals.

If a CRLP device is defined, the CIRB transaction can be run automatically by including it in the CICS startup jobstream. The CIRB statement should be coded just as it would if it were entered from a terminal. Include a slash (\) at the end of the statement to indicate the end of data. Figure 31 shows an example:

```
// EXEC DFHSIP,SIZE=NNNNK
CIRB PASSWORD,3,PRODCICS,0\
/*
```

Figure 31. Automatically Starting CIRB

If a sequential DASD device has been defined in the CICS DFHTCT, you must define two sequential DASD data sets: one input and one output. These can be either sequential access method (SAM) data sets or SAM-managed VSAM data sets. The input data set must contain the CIRB statement. (A utility such as DITTO or VSAM IDCAMS can be used to load the CIRB statement to the data set.) The output data set will contain the messages from the CIRB startup process. Whichever type of device is used -- CRLP or DASD -- do not include a CSSF GOODNIGHT statement following the CIRB statement, as this would allow the statement to be processed in all CICS startup modes (cold, auto, and emer).

The application server must be started before CICS for automatic startup to work. When the CIRB transaction successfully ends, the following message is displayed at the VSE console:

```
ARI0410I Resource Adapter ARI0OLRM is enabled
```

For more information about CICS sequential device support, see the *CICS Transaction Server for VSE/ESA V1R1.0 Resource Definition Guide* manual. For information about the DFHTCT entries required to define a sequential CRLP or DASD device, see the *DB2 Server for VM Program Directory*.

If a failure occurs, you can issue the CIRT transaction with the QUICK mode. This mode disconnects links to the application server. For more information, see "Stopping the Online Support -- The CIRT Transaction" on page 112. If the above action does not solve the problem, CICS must be recycled.

SCHEDULE Authority for VSE Guest Sharing

The VM database must grant SCHEDULE authority to the CICS/VSE application identifier.

Implicit CONNECT Support

This support allows development of online applications that do not issue an SQL CONNECT statement. With this support, operators need not enter a user ID and password as input to the online application, which is useful if your installation requires terminal users to sign on using the CSSN transaction. For some transactions accessing the database, the CICS sign-on verification may be sufficient. It can also be useful if you have just installed the database manager and find it convenient to have all users identified by one name (for example, CICSUSER).

If a CICS transaction has not yet established a user ID for the current or prior unit of work, and the user has signed on to CICS using the CESN (or CSSN) transaction, online support will attempt to use the eight-character sign on user ID. The user ID used will be the value returned by the CICS command

```
EXEC CICS ASSIGN USERID(data-area)
```

If you start the online support with CIRB, then before the online resource adapter is able to run the implicit connect support to a local application server, it verifies that the CICS subsystem has SCHEDULE authority on the local application server. Refer to procedure ARIS080D in the *DB2 Server for VSE Program Directory* manual, which shows how the CICS subsystem is identified to the application server and granted the necessary SCHEDULE authority. Modify the example procedure ARIS080D if any of the following are true:

- Your CICS subsystem does not use the default APPLID (DBDCCICS). It specifies, for example, APPLID=CICSTEST for the CICS DFHSIT.
- You want to change the password. For example, you want to identify the name by a more secure password than SQLDBAPW.

Given the above two conditions, you would change the GRANT statement to read: Grant the necessary SCHEDULE authority as follows:

```
GRANT SCHEDULE TO CICSTEST IDENTIFIED BY cicspw
```

where *cicspw* is the new password. The required password input parameter for CIRB (and CIRT) is now *cicspw*.

If the online support can verify that the CICS subsystem has SCHEDULE authority, it sets the DEFUID into each of the agents allocated for online use. The DEFUID you specify as an input parameter for CIRB is the user ID used for all online applications connecting to a local application server that do not issue an SQL CONNECT statement and do not have a valid CICS sign on user ID.

Supporting Multiple User Online Access

The NOLINKS input parameter to CIRB causes the allocation of a fixed number of links to the local application server. The online support suballocates the links to CICS transactions when they issue their first SQL request. When a transaction has a link, it keeps it until the end of the logical unit of work. When the number of such transactions exceeds NOLINKS, some transactions have to wait for links, and link contention occurs. Some planning is required to optimize the NOLINKS parameter. NOLINKS varies as your application mix varies.

Consider these things about the NOLINKS input parameter:

- Initially, allow one link for each one to two ISQL users, and one link for each four to ten users of preplanned transactions.
- The NOLINKS value must not exceed that of the NCUSERS initialization parameter, which defines the total number of links to the application server.
- The online support uses the CICS monitoring facility to collect performance data. For a given NOLINKS and a given period of the day, you can gather information on the number of link waits, total link wait time, and total time holding the link. For more information, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

- When a logical unit of work ends, the online support makes the freed link available to all waiting transactions. The first waiting database transaction that CICS dispatches gets the link. To define allocation priority for the online links, consider using the operator, transaction, and terminal priority mechanisms of CICS. (These are specified with the OPPRTY keyword of DFHSNT, and the TRMPRTY keyword of DFHTCT respectively.)
- Consider defining one or more transaction classes for the transactions that access the database manager, and limit access by using the CICS CMXT keyword of DFHSIT. By correlating CMXT with NOLINKS, you can ensure that storage resources in the CICS partition are not used until links are available.
- Consider a similar technique to control the number of active ISQL users. Rather than limit the total number of active ISQL users, you can control the number of active users from a given department or user group. See "Controlling Access by ISQL Users" on page 119.

CIRB Impact to System Resources

If the NOLINKS input parameter is *n*, system resources are used as follows:

- You have *n* links allocated to the application server, and *n* application server agents are used. The agents remain allocated for online applications until CIRT is entered.
- Additional virtual storage is required in the CICS partition for the online support. See Appendix A, "Processor Storage Requirements," on page 339.
- For each concurrent transaction that is attempting to access the application server, additional virtual storage is required in the CICS partition. See Appendix A, "Processor Storage Requirements," on page 339.

Supporting Multiple CICS Partitions

Your installation can have multiple CICS partitions, each with access to the application server. For recovery purposes, each instance of an active online resource adapter must have a unique identifier. You can do this with the CIRB RMID input parameter. You should keep the RMID for a CICS partition consistent, by relating the RMID to the priority of each CICS, specifying a `0` for the production CICS, 1 for the test-level CICS, and so on. If your installation has only one CICS system, the RMID input parameter need not be specified.

Adding Connections -- The CIRA Transaction

The CIRA transaction has four parameters:

```
►►──CIRA───┬─,──────────┬──┬─,─────────┬──┬─,────────┬──┬─server_name──────────────┬──►◄
           └─password,──┘  └─nolinks,──┘  └─defuid,──┘  │    ┌─,──────────────┐    │
                                                        │    ▼                │    │
                                                        └─(───server_name─────┴─)──┘
```

Figure 32. CIRA Transaction Syntax

The parameters are described in the following table:

Table 9. CIRA Transaction Parameters

Parameter	Default	Description

PASSWORD (positional parameter 1)	SQLDBAPW	This parameter establishes the operator's authority to activate online access to a local application server. The password identifies the CICS subsystem. The user ID of the subsystem is the CICS APPLID, which defaults to DBDCCICS. The procedure ARIS080D uses the following job control to give the password and user ID to the DB2 Server for VSE server:
		`// EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,`
		`LOGMODE=N,PROGNAME=ARIDBS'`
		`CONNECT SQLDBA IDENTIFIED BY SQLDBAPW;`
		`GRANT SCHEDULE TO DBDCCICS IDENTIFIED BY CICSPSWD;`
		`COMMIT WORK;`
		The password chosen (CICSPSWD above) must satisfy DB2 Server for VSE & VM specifications for a password. This password establishes which password to use when dropping connections through the CIRR or CIRT commands. See "Password Implications on Online Resource Adapter Termination" on page 116 for more details.

NOLINKS (positional parameter 2)	3	This parameter establishes the number of links (paths) that should be initialized to a local application server. Specify this parameter as a decimal value between 1 and 64. The number must be less than or equal to the value assigned to the NCUSERS initialization parameter of the DB2 Server for VSE & VM system. (The NCUSERS default is 5).
DEFUID (positional parameter 3)	CICSUSER	This parameter identifies the default user ID used by the online support when it makes an implicit CONNECT to a local application server. This parameter must satisfy DB2 Server for VSE & VM specifications for a user ID.
SERVER-NAME (positional parameter 4)	none	This parameter is required and it specifies the additional application servers (local or remote), that you want to access. If this parameter is omitted, the message ARI0400E is issued indicating that an invalid input parameter was entered.

The password, nolinks, defuid and server_name parameters have exactly the same meanings as on the CIRB command. One exception is that the server_name parameter is required on CIRA but is optional on CIRB.

The number of server_names that can be specified on the CIRA command is limited by the size of the input line. As with CIRB, CIRA can be entered on the VSE console or on a CICS terminal. On the VSE console the input is limited to one line. On the CICS terminal it can use the full screen. If short server_names are used more can fit on the command. Server_names can be up to 18 characters long. If all of the required server_names cannot fit on the command, the CIRA transaction must be repeated for the remaining server_names. Figure 33 on page 92 shows an example using the CIRA transaction with a server_name list.

```
msg f2
AR 015 1I40I READY
2 cirb ,,,,,sqlmach1
F2-002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2-002 ARI0450I DB2 Server for VSE online support has an
                entry point of 003AA808 RMGL at 00541200.
F2-002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0055A2E0.
F2-002 ARI0458I The default server is SQLMACH1.
2 cira ,,,(sqlmach2,sqlvm)
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055C2E0.
F2-002 ARI0454I Connections to SQLVM established.
                RMCV at 0055D2E0.
```

Figure 33. Example of CIRA with Server_Name List

The maximum number of application servers to which an online resource adapter can establish connections or enable online access to is only limited by the amount of storage available in the partition where the online resource adapter is running.

The CIRA transaction establishes connections or enables online access to the specified application servers based on the parameters given on the CIRA transaction. If a server_name list is used then connections or online access will be established to each application server in the list using the same set of parameters. For example:

```
CIRA thispw,4,thisid,(sqlmach2,sqlvm)
```

The above command will establish four links to the local application server SQLMACH2 with password *"thispw"* and default user ID *"thisid."* The RMID and the language ID are inherited from the CIRB transaction. If the online resource adapter was started with RMID = 0 and language ID = ameng then any connections started to that same online resource adapter will also have RMID = 0 and language id = ameng. Then CIRA will establish four links to SQLVM with password *"thispw"* and default user ID *"thisid."* Again the RMID is 0 and the language ID is ameng. If CIRA is entered before CIRB was run, the message *"ARI0411I Resource Adapter is not enabled."* is displayed.

If one or more of the parameters must be different, then the server_name list format of the CIRA transaction cannot be used. The CIRA transaction would have to be executed separately for each application server that required different parameters. For example, if three links are required to SQLMACH2 and four links are required to SQLVM but the other parameters are the same for both servers, the CIRA transaction must be run for each of them.

```
CIRA thispw,3,thisid,sqlmach2
CIRA thispw,4,thisid,sqlvm
```

If you try to establish connections or enable online access to an application server that is already connected a warning message will be displayed. No action is taken against that server. If the connections to a local application server need to be changed they must first be removed using CIRR or CIRT and then re-established using CIRA or CIRB.

Consider the following scenario. An online transaction program needs to access three different application servers, SQLMACH2, SQLMACH1 and SQLVM. SQLMACH2 and SQLMACH1 are running in two VSE partitions and SQLVM is running under VM and is accessed via guest sharing. We want SQLMACH1 to be the default server, and we want the default settings for all three servers.

To achieve this we could enter the following sequence of commands. Assume that our CICS region is running in partition 2, SQLMACH2 is running in partition 4 and SQLMACH1 is running in partition 5.

1. Use the CIRB transaction to start the online resource adapter and establish the default application server, SQLMACH1.
2. Use the CIRA transaction to establish connections to SQLMACH2.
3. Use the CIRA transaction again to establish connections to SQLVM.

This is illustrated in Figure 34.

```
F2-002 DFH1500 - DBDCCICS : CONTROL IS BEING GIVEN TO CICS
msg f2
AR 015 1I40I READY
2 cirb ,,,,,sqlmach1
F2-002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2-002 ARI0450I DB2 Server for VSE online support has an
                entry point of 003AA808 RMGL at 00541200.
F2-002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0055D2E0.
F2-002 ARI0458I The default server is SQLMACH1.
2 cira ,,,sqlmach2
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055C2E0.
2 cira ,,,sqlvm
F2-002 ARI0454I Connections to SQLVM established.
                RMCV at 0055A2E0.
```

Figure 34. Example of CIRB and CIRA

Since the settings for the connections to SQLMACH2 and SQLVM are identical, both connections could be established on the same CIRA command, as illustrated in Figure 33 on page 92.

Automatic Restart Resynchronization

If a system or subsystem failure occurs while an online application is trying to commit work and two-phase commit is being used, the unit being committed is called an in-doubt logical unit of work, because the database manager has prepared it for commit or rollback but the system or subsystem failure occurred before the commit completed. In-doubt units of work must be resolved the next time the application server is started.

Note: CICS/VSE and the local application server will use a one-phase commit if at most one external resource has been updated. In this case it is not possible to create an in-doubt unit of work. This means that any CICS transaction that updates only the local application server resources will not generate in-doubt units of work.

The CICS/VSE restart resynchronization facility, which is started implicitly when you issue CIRB or CIRA, resolves the in-doubt units of work created by any CICS transaction that updated a local application server. To enable it, you must update the CICS/VSE tables to include the resynchronization transaction.

CIRB and CIRA assume that restart resynchronization is enabled when they are executed. If, for some reason it has been disabled when CIRB or CIRA is issued, it will display the message *"ARI0466E CICS restart re-synchronization is not available. The <tran> transaction is ended."* and exit. At this point the system programmer should ensure that it has been properly enabled and retry CIRB or CIRA.

For information about the updates, see the *DB2 Server for VSE Program Directory* manual.

The current implementation of the CICS/VSE restart resynchronization facility allows it to re-synchronize itself with DB2 Server for VSE online resource adapter only once. After it has been invoked, CICS discards any information about in-doubt units of work that it did not resolve. This means that there can be scenarios where it is not possible to automatically resolve in-doubt units of work.

When the CIRB or CIRA transaction is started, a connection is made to the READY/RECOVERY agent of the local server to get a 'recovery list'. This recovery list provides information on any in-doubt agents that need to be resolved for this server. After this has been done for every local server specified in the CIRB or CIRA command, the CICS/VSE restart resynchronization facility is invoked, which will resolve the in-doubt units of work for all of those local servers. A subsequent CIRA to connect to another local server that also has in-doubt units of work will fail because CICS has discarded the log information. The in-doubt units of work on that server must be resolved manually using the FORCE n COMMIT or FORCE n ROLLBACK commands on the server before the CIRA command will work.

For example, suppose that SQLMACH1 and SQLMACH2 are DB2 Server for VSE application servers that run on the same VM system and are accessed via guest sharing. The password used to access SQLMACH1 is ABC and the password used to access SQLMACH2 is DEF. All the other parameters needed by the two databases are the defaults. The connections to SQLMACH1 and SQLMACH2 are established using the following sequence of commands:

```
CIRB abc,,,,,sqlmach1

CIRA def,,,sqlmach2
```

Suppose that CICS transactions accessing these application servers also make updates to the DB2 Server for VSE database as well as some other external non-CICS resource, so that CICS will use the two-phase commit process. If a system failure occurs on the VM system while CICS is performing a two-phase commit to both these databases, then both SQLMACH1 and SQLMACH2 will go down. When the system is brought back up and SQLMACH1 and SQLMACH2 are restarted, they will both have in-doubt units of work. If the connections to SQLMACH1 and SQLMACH2 are restarted the same way as before, only the in-doubt units of work on SQLMACH1 will be resolved automatically. The in-doubt units of work on SQLMACH2 will need to be resolved explicitly before the CIRA command for SQLMACH2 will work.

See Figure 35 on page 95 for an example of this.

```
2 cirb abc,,,,,sqlmach1
F2 002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2 002 ARI0450I DB2 Server for VSE online support has an
                entry point of 0039F008 RMGL at 001DF5B4.
F2 002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0053BF00.
F2-002 ARI0458I The default server is SQLMACH1.
2 cira def,,,sqlmach2
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055A080.

<System Failure occurs>

F2 002 ARI2908I XPCCB, IJBXRUSR = 0483061009000000
F2 002 ARI0406E Error in using system communications facility.
                Request = 15
                Return Code = 4 Reason Code = 7
F2 002 The default server is SQLMACH1.
F2 002 ----------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH1.
F2 002 Status of DB2 Server for VSE online applications:
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO  TRANID TERMID USER ID   USERDATA TIME SINCE  TOTAL LUW
F2 002                                           LAST ACCESS TIME
F2 002  ______  ______ ______ ________  ________ ___________ _________
F2 002  0000041  CISQ         SQLDBA     L080    00:00:06    00:01:34
F2 002
F2 002  TIME= 15:26:15 DATE= 08/14/95
F2 002 ARI0465I Transactions are still active
                for server SQLMACH1.
F2 002 ARI0463I The DISABLE transaction CIRR must delay for a
                30-second interval before attempting the disable.
F2 002 ARI0455I Connections to SQLMACH1 are disabled.
F2 002 ARI0460W Connections to the default server SQLMACH1
                have been disabled.
F2 002 ARI2908I XPCCB, IJBXRUSR = 0483061009000000
F2 002 ARI0406E Error in using system communications facility.
                Request = 15
                Return Code = 4 Reason Code = 7
F2 002 The default server is SQLMACH1.
F2 002 ----------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH2.
F2 002 Status of DB2 Server for VSE online applications:
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO  TRANID TERMID USER ID   USERDATA TIME SINCE  TOTAL LUW
F2 002                                           LAST ACCESS TIME
F2 002  ______  ______ ______ ________  ________ ___________ _________
F2 002  0000141  CISQ         SQLDBA     L083    00:00:06    00:01:34
F2 002
```

Figure 35. Automatic Restart Resynchronization Failure (Part 1 of 2)

```
F2 002  TIME= 15:26:45 DATE= 08/14/95
F2 002 ARI0465I Transactions are still active
                for server SQLMACH2.
F2 002 ARI0463I The DISABLE transaction CIRR must delay for a
                30-second interval before attempting the disable.
F2 002 ARI0455I Connections to SQLMACH2 are disabled.
F2-002 ARI0413I Resource Adapter ARI0OLRM is disabled.

<SQLMACH1 and SQLMACH2 are restarted>

2 cirb abc,,,,,sqlmach1
F2 002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2 002 ARI0450I DB2 Server for VSE online support has an
                entry point of 0039F008 RMGL at 001DF5B4.
F2 002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0053BF00.
F2-002 ARI0458I The default server is SQLMACH1.
2 cira def,,,sqlmach2
F2 002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055A080.
F2-002
F2 002 ARI0438E Automatic restart resynchronization failed.
                A logical unit of work that DB2 for VSE indicated
                needed to be resolved was not identified by
                the CICS/VSE log as needing resolution.
F2 002 ARI0423A Use the SHOW and FORCE commands to
                COMMIT or ROLLBACK the following units of work:
F2 002 ARI0424I User ID = SQLDBA Agent Identifier = 1
                Server = SQLMACH2
F2 002 The default server is SQLMACH1.
F2 002 ---------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH2.
F2 002 There are no active DB2 Server for VSE transactions.
F2 002
F2 002  TIME= 15:33:22 DATE= 08/14/95
F2 002 ARI0455I Connections to SQLMACH2 are disabled.

<From the SQLMACH2 console enter:>
<SHOW ACTIVE>
<FORCE 1 ROLLBACK>

<Now CIRA will work>

2 cira def,,,sqlmach2
F2 002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055A080.
```

Figure 35. Automatic Restart Resynchronization Failure (Part 2 of 2)

However if the connections to SQLMACH1 and SQLMACH2 are established with a single CIRB or CIRA command, the in-doubt units of work on **both** servers will be resolved automatically.

See Figure 36 on page 97 for a detailed example of this.

```
2 cirb abc,,,,,(sqlmach1,sqlmach2)
F2 002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2 002 ARI0450I DB2 Server for VSE online support has an
                entry point of 0039F008 RMGL at 001DF5B4.
F2 002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0053BF00.
F2-002 ARI0458I The default server is SQLMACH1.
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055A080.

<System Failure occurs>

F2 002 ARI2908I XPCCB, IJBXRUSR = 0483061009000000
F2 002 ARI0406E Error in using system communications facility.
                Request = 15
                Return Code = 4 Reason Code = 7
F2 002 The default server is SQLMACH1.
F2 002 ---------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH1.
F2 002 Status of online DB2 Server for VSE applications:
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO  TRANID TERMID USER ID   USERDATA TIME SINCE  TOTAL LUW
F2 002                                           LAST ACCESS TIME
F2 002  ______  ______ ______ ________  ________ ___________ _________
F2 002  0000041  CISQ         SQLDBA     L080    00:00:06    00:01:34
F2 002
F2 002  TIME= 15:26:15 DATE= 08/14/95
F2 002 ARI0465I Transactions are still active
                for server SQLMACH1.
F2 002 ARI0463I The DISABLE transaction CIRR must delay for a
                30-second interval before attempting the disable.
F2 002 ARI0455I Connections to SQLMACH1 are disabled.
F2 002 ARI0460W Connections to the default server SQLMACH1
                have been disabled.
F2 002 ARI2908I XPCCB, IJBXRUSR = 0483061009000000
F2 002 ARI0406E Error in using system communications facility.
                Request = 15
                Return Code = 4 Reason Code = 7
F2 002 The default server is SQLMACH1.
F2 002 ---------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH2.
F2 002 Status of online DB2 Server for VSE applications:
F2 002
```

Figure 36. Successful Automatic Restart Resynchronization (Part 1 of 2)

```
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO  TRANID TERMID USER ID   USERDATA TIME SINCE  TOTAL LUW
F2 002                                           LAST ACCESS TIME
F2 002  ______  ______ ______ _________ ________ ___________ _________
F2 002  0000141  CISQ         SQLDBA     L083    00:00:06    00:01:34
F2 002
F2 002  TIME= 15:26:45 DATE= 08/14/95
F2 002 ARI0465I Transactions are still active
                for server SQLMACH2.
F2 002 ARI0463I The DISABLE transaction CIRR must delay for a
                30-second interval before attempting the disable.
F2 002 ARI0455I Connections to SQLMACH2 are disabled.
F2-002 ARI0413I Resource Adapter ARI0OLRM is disabled.

<SQLMACH1 and SQLMACH2 are restarted>

2 cirb abc,,,,,(sqlmach1,sqlmach2)
F2 002 ARI0410I Resource Adapter ARI0OLRM is enabled.
F2 002 ARI0450I DB2 Server for VSE online support has an
                entry point of 0039F008 RMGL at 001DF5B4.
F2 002 ARI0454I Connections to SQLMACH1 established.
                RMCV at 0053BF00.
F2-002 ARI0458I The default server is SQLMACH1.
F2 002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055A080.
```

Figure 36. Successful Automatic Restart Resynchronization (Part 2 of 2)

Assuming CICS restart resynchronization has been properly enabled as described in the *DB2 Server for VSE Program Directory* manual, the conditions where in-doubt units of work must be resolved explicitly are:

1. CICS log missing. This can be from a CICS log media failure, CICS COLD start which destroys the log contents, or CICS journal is not active so no log data is created.
2. CICS RESYNCH has already been issued. The log data is discarded by CICS after the RESYNCH command has been issued even if it was not used. See Figure 35 on page 95 for an example of this.

To take full advantage of the automatic restart resynchronization the following should be true:

1. All local application servers with in-doubt units of work must be started on the same CIRB or CIRA transaction. This means they must have the same password, default user ID, language, RMID, and number of links to be started.
2. CICS startup should be START=AUTO which lets CICS determine if the startup will be START=WARM or START=EMER. Any COLD start will erase the log data and automatic restart resynchronization will not be possible.

Resolving In-Doubt Transactions

Only under exceptional conditions (such as a CICS log media failure) do you have to resolve in-doubt LUWs explicitly. To do so, issue the `SHOW ACTIVE` command to determine those agents that are in-doubt; then issue the `FORCE` command to commit or rollback each one:

```
FORCE n COMMIT

  or

FORCE n ROLLBACK
```

where *n* is the agent identifier of the in-doubt LUW.

The discussion in the *DB2 Server for VSE & VM Operation* manual states that, in general, FORCE *n* COMMIT should be entered. The exception is for applications that access multiple resources (for example, an application that updates a database and a VSAM file.) For such applications, the operator requires direction from the developer or user of the application.

You could plan for this situation by keeping a list of all transactions that update multiple resources. The list should contain the CICS transaction identifier for the application, and the recommended direction (COMMIT or ROLLBACK) from the developer. (For more information, see the discussion on online application recovery in the *DB2 Server for VSE & VM Database Administration* manual.) Because ISQL does not update multiple resources, the direction for the ISQL transaction should always be to commit work.

Changing the Default Server -- The CIRC Transaction

The transaction CIRC can be used to dynamically change the default server. The CIRC transaction has one parameter:

```
►►──CIRC──server_name──────────────────────────────────────────────►◄
```

Figure 37. CIRC Transaction Syntax

The parameter is described in the following table.

Table 10. CIRC Transaction Parameter

Parameter	Default	Description
SERVER-NAME (positional parameter 1)	none	This parameter is required and it specifies the application server that you want to become the default. If this parameter is omitted, the message ARI0400E is issued indicating that an invalid input parameter was entered.

The server-name specified must already have connections or online access established to it, either from the CIRB or CIRA transactions. If connections to the specified server do not exist or online access to the specified server was not enabled from the CIRB or CIRA transactions, the message *"ARI0456I Connections to <server-name> do not exist."* is displayed. In this case the CIRA transaction must first be run to establish the connections, then the CIRC transaction is run to make it the default server.

For the following example assume that connections exist to SQLMACH1 and SQLMACH2 and that SQLMACH2 is the current default server.

```
msg f2
AR 015 1I40I READY
2 circ sqlmach1
F2-002 ARI0459I The new default server is SQLMACH1.
                The previous default server was SQLMACH2.
```

Figure 38. Example of CIRC

For this next example assume that connections exist to SQLMACH1 but not to SQLMACH2.

```
msg f2
AR 015 1I40I READY
2 circ sqlmach2
F2-002 ARI0456I Connections to SQLMACH2 do not exist.
2 cira ,,,sqlmach2
F2-002 ARI0454I Connections to SQLMACH2 established.
                RMCV at 0055D2E0.
2 circ sqlmach2
F2-002 ARI0459I The new default server is SQLMACH2. The previous
                     default server was SQLMACH1.
```

Figure 39. Example of CIRC

It is important to note that if the connections to the default server are lost, or online access to the default application server is disabled, that server is still identified as the default server. The connections can be lost because the server went down or because the CIRR transaction was used to terminate the online access or connection. Users that are trying to connect to the default server in these cases will receive SQLCODE = -940. If the CIRB or CIRA transaction is used to establish connections to a local server that is not ready, the message *"ARI0418A Application server <server-name> is not ready. Retry the enable transaction after the application server starts."* is displayed. If the CIRB or CIRA transaction is used to establish online access to a remote server that is not ready, an error message will not be displayed. This is because CIRB or CIRA can not check whether a remote server is ready or not.

Removing Connections -- The CIRR Transaction

To remove connections or to disable online access to a local or remote application server, issue the CICS CIRR transaction. The CIRR transaction has four parameters:

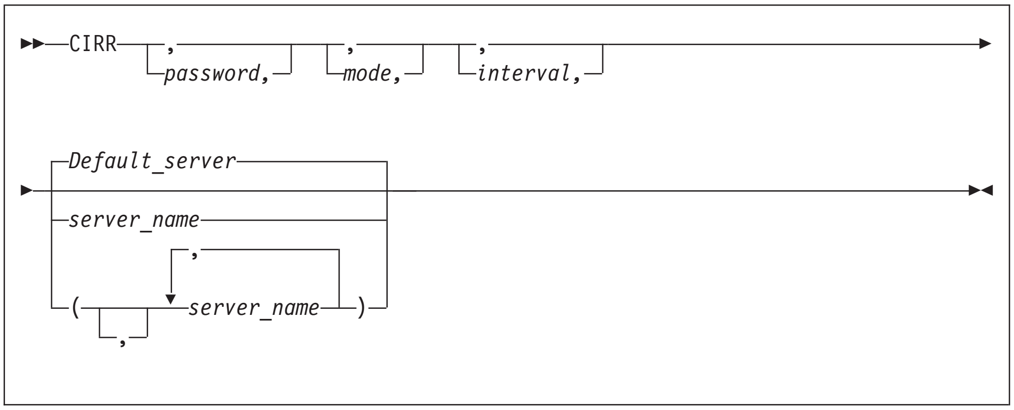


Figure 40. CIRR Transaction Syntax

The password, mode and interval parameters are the same as on the CIRT transaction and are described in the following table:

Table 11. CIRR Transaction Parameters

Parameter	Default	Description
PASSWORD (positional parameter 1)	SQLDBAPW	This password establishes the operator's authority to terminate the online access to the application server. It must be the same password that was supplied for the server by the CIRB or CIRA transaction. Refer to "Password Implications on Online Resource Adapter Termination" on page 116 for more details.
MODE (positional parameter 2)	NORMAL	This parameter establishes the shutdown mode: NORMAL or QUICK. When you specify NORMAL, the CIRR transaction prevents new online users from accessing the specified application server. Users who are already doing work, however, can finish. When all users complete their work, no online users can use the specified application server. When you specify NORMAL for a remote application server, the shutdown of the access to the remote application server will complete only when all conversations to the remote application server have been deallocated. When you specify QUICK for a local application server, online access is ended immediately. Online users cannot finish their work. Their current logical units of work are rolled back (unless they are already processing a COMMIT WORK). You can change from NORMAL to QUICK. However, once the MODE is QUICK, you cannot change it back to NORMAL. When you specify QUICK for a remote server, the QUICK mode is changed to NORMAL. QUICK mode is not supported for a remote application server.

Table 11. CIRR Transaction Parameters (continued)

Parameter	Default	Description
INTERVAL (positional parameter 3)	30 (seconds)	The number of seconds that the CIRR transaction should delay before freeing the terminal. The value must be an integer value between 0 and 3600. This parameter controls the availability of the CICS terminal (or operator console) once you issue the CIRR transaction. The CICS terminal (or VSE operator console) used to activate the CIRR transaction is unavailable until the transaction ends. This could be a long time if the online application is long-running or if a user left without correctly ending the terminal session. If you issue CIRR PASSWORD,NORMAL,, server_name the terminal is not available until all online DB2 Server for VSE users complete their work. The value you specify for interval represents an interval of time measured in seconds. If the CIRR transaction does not finish immediately, it waits the amount of time you specify. When this time ends, the CIRR transaction tries once again to finish processing. If the CIRR transaction does not finish successfully, you receive a message telling you to retry the CIRR transaction later. After issuing the message, the CIRR transaction ends. The shutdown mode is still in effect (the specified server is in the process of shutting down), and the terminal is available for your use.
SERVER-NAME (positional parameter 4)	Determined by CIRB or CIRC transaction.	This parameter enables you to specify the application servers from which you want to remove access. The default server is removed if this parameter is omitted, or if the first parameter in the server_name list is blank. The default server is the one that was established by the CIRB transaction or by the CIRC transaction.

If no server_name is specified the default server_name is used. The default server_name was established by the CIRB or CIRC transaction. The CIRD transaction may be used to display the default server_name in case the user does not know what the default server_name is.

```
msg f2
AR 015 1I40I READY
2 cirr
F2-002 ARI0455I Connections to SQLMACH1 are disabled.
F2-002 ARI0460W Connections to the default server SQLMACH1 have
                     been disabled.
```

Figure 41. Example of CIRR with Defaults

The above example assumes that there are connections to more than one server when the CIRR transaction is entered.

If the password, mode and interval are the same then the server_name list can be used to remove connections or disable online access from multiple application servers. Since SQLVM was the last active connection, the online resource adapter was terminated. SQLMACH2 and SQLVM are local application servers, while SQLMACH8 is a remote server.

```
msg f2
AR 015 1I40I READY
2 cirr ,,,(sqlmach2,sqlmach8,sqlvm)
F2-002 ARI0455I Connections to SQLMACH2 are disabled.
F2-002 ARI0455I Online access to SQLMACH8 is disabled.
F2-002 ARI0455I Connections to SQLVM are disabled.
F2-002 ARI0413I Resource Adapter ARI0OLRM is disabled.
```

Figure 42. Example of CIRR with Server-Name List

The CIRR transaction can be used to remove the connections or disable online access to the application server that were established by the CIRB and CIRA transactions. If CIRR removes the last active connections to the online resource adapter and all active APPC conversations known to the online resource adapter are deallocated, then the online resource adapter is terminated. The CIRB transaction would have to be used to restart it.

The CIRA and CIRR transactions can be entered repeatedly and in any order to add and remove links to application servers or to enable and disable online access to application servers as required.

If CIRR is entered to remove connections or disable online access to a server to which no connections or online access have been established, the message *"ARI0456I Connections to <server_name> do not exist."* is displayed.

If the password given on the CIRR transaction does not match the password that was used to start the connections or online access to the named server, then the connections or online access to that server are not shut down and processing continues with the next server in the list.

Displaying Information -- The CIRD Transaction

To display status information about active CICS transactions that access a local or a remote application server, issue the CICS CIRD transaction.

The CIRD transaction does not require a password, and can be issued from any CICS terminal or the operator console. To use it, you must enable it as well as the CICS restart resynchronization facility. See the *DB2 Server for VSE Program Directory* for more information.

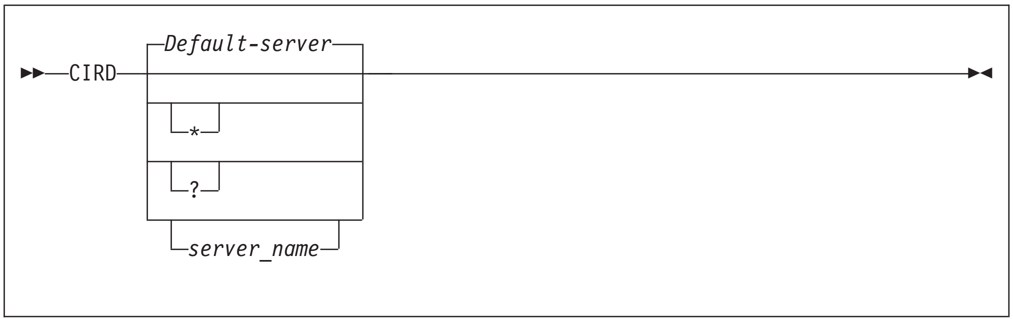


Figure 43. CIRD Transaction Syntax

The parameter is described in the following table:

Table 12. CIRD Transaction Parameters

Parameter	Default	Description
SERVER-NAME (positional parameter 1)	Determined by CIRB or CIRC transaction.	This parameter enables you to specify the application server whose status is to be displayed, or * to display the status of all servers and the details of transactions accessing the servers, or ? to display a list of the connected servers without the transaction details. If this parameter is omitted, the default server_name is the one that was determined by the CIRB or the CIRC transaction.

Four categories of CICS transactions access the local application server. The information that CIRD displays for transactions connected to a local server varies depending on these four categories:

- Transactions waiting to access the local application server

  These transactions have issued an SQL request and are waiting because all links to the application server are busy. For these transactions, CIRD displays the elapsed time of the wait.

  In general, links to the local application server are busy because other users are accessing it. The only exception occurs when the DB2 Server for VSE online support is being started; at that time, all links to the application server could be busy during the synchronization of the database log and the CICS log. Usually this requires little time, but a long delay can occur if a very large LUW is being rolled back.
- Transactions currently accessing the local application server

  These transactions have established a link to the local application server and an LUW. The application server is currently doing processing for that LUW. For these transactions, CIRD displays the elapsed time of the current SQL statement, and the elapsed time the link is held. The latter effectively indicates the elapsed time of the current LUW.
- Transactions holding a link to the local application server but not using it

  These transactions have established a link to the local application server and an LUW, but the application server is not currently processing for that LUW. Instead, these transactions are doing other work or are waiting for terminal communications. For these transactions, CIRD displays the elapsed time since the last application server access ended, and the elapsed time the link is held. Again, the latter effectively indicates the elapsed time of the current LUW.
- Transactions that previously held a link to the local application server, but no longer do.

  These transactions have previously ended one or more LUWs, but have not yet started another. For these transactions, CIRD displays the elapsed time since the last LUW completed.

If you enter CIRD when the DB2 Server for VSE online support is not enabled or when the CIRD is not operational, an error message is displayed and CIRD ends. Note that for CIRD to display information about a transaction, the transaction must have issued an SQL request. CIRD displays the following information (where applicable) for each of the four categories of local database transactions:

- The CICS task number (TASKNO)

- The CICS transaction identifier (TRANID)
- The CICS terminal identifier (TERMID)

  Not all transactions have a terminal identifier. For example, ISQL has a two-transaction structure: ISQL and CISQ. The former controls the terminal and the latter is for access to the application server. Because a CISQ transaction has no terminal associated with it, instead of displaying TERMID for it, CIRD displays the terminal identifier in another field called USERDATA (described below).

  If a transaction accesses the application server, but does not have a terminal associated with it, CIRD does not display TERMID.
- The user identifier (USERID) that the application server establishes for the transaction

  CIRD does not display this identifier unless a user ID has been established, which is done when an application issues an SQL statement that starts an initial LUW. The user ID may not be established immediately. (For example, a transaction can be waiting for a link to the application server.) It remains established after a transaction ends an LUW, unless the RELEASE option of COMMIT WORK or ROLLBACK WORK was used.
- User data (USERDATA) for ISQL transactions

  The USERDATA field contains the terminal identifier (TERMID) of the terminal that was used to call ISQL. For most other transactions, USERDATA is blank. It is possible, however, to code an online application to initialize the USERDATA field. Such an application would use the DB2 Server for VSE online cancel support. For more information, see "Coding Your Own Cancel Exit" on page 287.

  Note: If you are controlling ISQL access with the DFHSIT CMXT parameter, you have renamed the ISQL transaction. For these renamed ISQL transactions, CIRD still displays the terminal identifier of the terminal that was used to run the transaction. For more information on this parameter, see "Controlling Access by ISQL Users" on page 119.
- The elapsed time intervals (as described above)

  CIRD uses the following format to display the time:

  ```
  hh:mm:ss
  ```

CIRD then displays the time of day and the date, as follows:

```
TIME=hh:mm:ss DATE=mm/dd/yy   (or dd/mm/yy)
```

and then ends its processing. (The format of the date depends on how you specified it on the DATE parameter of the VSE STDOPT JCC/JCS.)

If CIRD determines that no CICS transactions apply to the application server, it displays only the time and the date, and then ends.

Note: If the DB2 Server for VSE online support ends abnormally (for example, if the application server partition ends unexpectedly), the CIRD transaction is called implicitly to display information about transactions that were accessing the application server at the time of the failure. This information is displayed on the VSE system console.

For the following examples, assume that SQLMACH1 is the default local server and that connections have been established for the local application servers SQLMACH1, SQLMACH2 and SQLVM.

Figure 44 shows an example of the information displayed by the CIRD transaction with no parameters.

```
2 cird
F2 002 The default server is SQLMACH1.
F2 002 ---------------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH1.
F2 002 Status of online DB2 Server for VSE applications:
F2 002
F2 002 Transactions waiting to establish a link to the application server are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA WAIT TIME
F2 002  ------ ------ ------ -------- -------- ---------
F2 002  000033 MKE2                   L222     00:01:32
F2 002  000025 INV    L224   JIM               00:08:32
F2 002
F2 002 Transactions holding a link and now accessing the application server are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME USED    TOTAL LUW
F2 002                                         FOR CURRENT  TIME
F2 002                                         ACCESS
F2 002  ------ ------ ------ -------- -------- ------------ ---------
F2 002  000019 CISQ          DEPT222  L199     00:01:32     00:03:48
F2 002  000037 INV    L209   TERRY             00:00:01     00:00:03
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME SINCE   TOTAL LUW
F2 002                                         LAST ACCESS  TIME
F2 002  ------ ------ ------ -------- -------- ------------ ---------
F2 002  000003 CISQ          WILLIAM  L210     00:07:01     00:10:56
F2 002
F2 002 Transactions which previously accessed the application server (not holding link):
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME SINCE
F2 002                                         LAST ACCESS
F2 002  ------ ------ ------ -------- -------- ------------
F2 002  000003 MKE2          ROBERT   L210     00:20:04
F2 002
F2 002  TIME=14:28:23 DATE=09/01/95
```

Figure 44. Example of CIRD with Defaults

Figure 45 on page 107 shows an example of the information displayed by the CIRD transaction with a server_name specified.

```
2 cird sqlmach2
F2 002 The default server is SQLMACH1.
F2 002 ---------------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH2.
F2 002 Status of online DB2 Server for VSE applications:
F2 002
F2 002 Transactions waiting to establish a link to the application server are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA WAIT TIME
F2 002  ------ ------ ------ -------- -------- ---------
F2 002  000033 MKE2                   L222     00:01:32
F2 002  000025 INV    L224   JIM               00:08:32
F2 002
F2 002 Transactions holding a link and now accessing the application server are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME USED    TOTAL LUW
F2 002                                         FOR CURRENT  TIME
F2 002                                         ACCESS
F2 002  ------ ------ ------ -------- -------- ------------ ---------
F2 002  000019 CISQ          DEPT222  L199     00:01:32     00:03:48
F2 002  000037 INV    L209   TERRY             00:00:01     00:00:03
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME SINCE   TOTAL LUW
F2 002                                         LAST ACCESS  TIME
F2 002  ------ ------ ------ -------- -------- ------------ ---------
F2 002  000003 CISQ          WILLIAM  L210     00:07:01     00:10:56
F2 002
F2 002 Transactions which previously accessed the application server (not holding link):
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME SINCE
F2 002                                         LAST ACCESS
F2 002  ------ ------ ------ -------- -------- ------------
F2 002  000003 MKE2          ROBERT   L210     00:20:04
F2 002
F2 002  TIME=14:28:23 DATE=09/03/95
```

Figure 45. Example of CIRD with Server-Name

Figure 46 on page 108 shows an example of the information displayed by the CIRD transaction with the * specified.

```
2 cird *
F2 002 The default server is SQLMACH1.
F2 002 There are connections to server SQLMACH1.
F2 002 There are connections to server SQLMACH2.
F2 002 There are connections to server SQLVM.
F2 002 ---------------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH1.
F2 002 Status of online DB2 Server for VSE applications:
F2 002
F2 002 Transactions waiting to establish a link to the application server are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA WAIT TIME
F2 002  ------ ------ ------ -------- -------- ---------
F2 002  000033 MKE2                   L222     00:01:32
F2 002  000025 INV    L224   JIM               00:08:32
F2 002
F2 002 Transactions holding a link and now accessing the application server are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME USED    TOTAL LUW
F2 002                                         FOR CURRENT  TIME
F2 002                                         ACCESS
F2 002  ------ ------ ------ -------- -------- ------------ ---------
F2 002  000019 CISQ          DEPT222  L199     00:01:32     00:03:48
F2 002  000137 INV    L209   BOB               00:17:34     01:24:03
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME SINCE   TOTAL LUW
F2 002                                         LAST ACCESS  TIME
F2 002  ------ ------ ------ -------- -------- ------------ ---------
F2 002  000013 CISQ          LARRY    L210     00:03:01     00:11:36
F2 002
F2 002 Transactions which previously accessed the application server (not holding link):
F2 002
F2 002  TASKNO TRANID TERMID USER ID  USERDATA TIME SINCE
F2 002                                         LAST ACCESS
F2 002  ------ ------ ------ -------- -------- ------------
F2 002  000003 MKE2          LOUISA   L210     01:57:04
F2 002
F2 002  TIME=14:28:23 DATE=09/03/95
F2 002 ---------------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH2.
F2 002 There are no active DB2 Server for VSE transactions.
F2 002
F2-002  TIME= 14:29:47 DATE= 09/03/95
F2 002 ---------------------------------------------------
F2 002 DBDCCICS connected to server SQLVM.
F2 002 There are no active DB2 Server for VSE transactions.
F2 002
F2 002  TIME=14:30:23 DATE=09/03/95
```

*Figure 46. Example of CIRD with **

Figure 47 on page 109 shows an example of the information displayed by the CIRD transaction with the ? specified.

```
2 cird ?
F2 002 The default server is SQLMACH1.
F2 002 There are connections to server SQLMACH1.
F2 002 There are connections to server SQLMACH2.
F2 002 There are connections to server SQLVM.
F2 002 ---------------------------------------------------
```

Figure 47. Example of CIRD with ?

Some extra information can be derived from the displays. In Figure 50 on page 112 notice that SQLMACH1 is mentioned as the default server and on the next message that there are connections to SQLMACH1 also. It is possible, with the CIRR transaction, to remove the connections to SQLMACH1. The CIRD command would still show that the default server is SQLMACH1 but the message indicating there are connections to SQLMACH1 would not be displayed. In this scenario, users connecting to the default server would receive SQLCODE = -940 on the CONNECT statement. The CIRA transaction could be used to establish connections to SQLMACH1 again or the CIRC transaction could be used to change the default server to one of the other active servers. Either method allows CONNECT statements to access the default server.

If CIRR or CIRT has been issued to disconnect a server or to shut down the online resource adapter but cannot complete because there are still active transactions against the server, the CIRD transaction will show which transactions and which servers are affected.

Figure 48 on page 110 shows an example of the information displayed by the CIRD transaction with the ? parameter specified. The attempt to remove the connections to SQLMACH2 fails because there are still active transactions. Then the CIRD transaction is used to determine which transactions are still active. The user is found and asked to complete his work. When the CIRR command is retried it completes successfully and the connections to SQLMACH2 are shut down.

```
2 cird ?
F2 002 The default server is SQLMACH1.
F2 002 There are connections to server SQLMACH1.
F2 002 There are connections to server SQLMACH2.
F2 002 There are connections to server SQLVM.
F2 002 ---------------------------------------------------
2 cirr ,,1,sqlmach2
F2 002 ARI0463I The DISABLE transaction CIRR must delay for a
                1-second interval before attempting the disable.
F2-002
2 cird ?
F2 002 The default server is SQLMACH1.
F2 002 There are connections to server SQLMACH1.
F2 002 Connections to SQLMACH2 are being disabled.
F2 002 There are connections to server SQLVM.
F2 002 ----------------------------------------------
F2-002
2 cird *
F2 002 The default server is SQLMACH1.
F2 002 There are connections to server SQLMACH1.
F2 002 Connections to SQLMACH2 are being disabled.
F2 002 There are connections to server SQLVM.
F2 002 ----------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH1.
F2 002 There are no active DB2 Server for VSE transactions.
F2 002
F2 002  TIME= 19:07:43 DATE= 09/20/95
F2-002
F2 002 ----------------------------------------------
F2 002 DBDCCICS connected to server SQLMACH2.
F2 002 Status of online DB2 Server for VSE applications:
F2 002
F2 002 Transactions holding a link to the application server but not using are:
F2 002
F2 002  TASKNO  TRANID TERMID USER ID  USERDATA TIME SINCE  TOTAL LUW
F2 002                                          LAST ACCESS TIME
F2 002  ______  ______ ______ ________ ________ ___________ _________
F2 002  0000129 CISQ          CICSUSER L77D     00:00:31    00:00:31
F2 002
F2 002  TIME= 19:07:44 DATE= 09/20/95
F2 002 ----------------------------------------------
F2 002 DBDCCICS connected to server SQLVM.
F2 002 There are no active DB2 Server for VSE transactions.
F2 002
F2 002  TIME= 19:07:45 DATE= 09/20/95
F2-002
2 cirr ,,2,sqlmach2
F2-002 ARI0455I Connections to SQLMACH2 are disabled.
```

Figure 48. Example of CIRD in a Disable Scenario

The CIRD transaction displays the following information (where applicable) for transactions that relate to a remote application server:

RDBMS
: displays the name, class, and release level (version, release, and modification level) of the application server being accessed.

LU
: displays the logical unit name.

TPN
: displays the transaction program name. Its character and hexadecimal versions are both displayed.

TASKNO
displays the number of the task.

TRANID
displays the transaction id.

TERMID
displays the name of the terminal where the transaction was initiated.

USER ID
displays the connected user id.

STATUS
displays the communication state. COMM indicates that the transaction sent an SQL statement to the database machine and has been waiting for a reply since the time shown. APPL indicates that the transaction returned control to the application at the time shown. VRA indicates that the Online Resource Adapter is processing your request. WAIT indicates that the transaction is waiting for a session.

TIME
displays the time when the STATUS displayed had begun. For example, task number 891 has already returned control to the application at 09:12:42, as indicated by TIME.

LUWID
displays the logical unit of work identifier, which uniquely identifies an LU6.2 conversation. Its value is *netid.luname.instance_number.sequence_number* where *netid* and *luname* are up to 8 characters long, *instance_number* is 12 characters long, and *sequence_number* is 4 characters long.

Figure 49 shows an example of the information displayed by the CIRD transaction with a remote server-name specified.

```
User:   2 cird sqlmach8
System: F2 0002 The default server is SQLMACH8.
        F2 0002 ----------------------------------------------
        F2 0002 Status of online DB2 Server for VSE applications for
        F2 0002 RDBMS = SQLMACH8 SQLDS/VM V6.1.0
        F2 0002 LU = VMC3
        F2 0002 TPN = SQLMACH8
        F2 0002       (X'E2D8D3D4C1C3C8F8')
        F2 0002
        F2 0002  TASKNO  TRANID TERMID USER ID   STATUS   TIME
        F2 0002  ______  ______ ______ ________  ______   ___________________
        F2 0002          LUWID
        F2 0002          ______
        F2 0002  0000891  DRT1   D080  SYSA       APPL    1998-08-11.09:12:42

        F2 0002           CAIBMOML.D08001.E31FE596ADDE.0001

        F2 0002
        F2 0002  TIME= 09:18:11 DATE= 08/11/98
        F2-0002
```

Figure 49. Example of CIRD with remote server name

Figure 50 on page 112 shows an example of the information displayed by the CIRD transaction with a ? specified, where online access to the remote server RMTSERV1 is allowed. Assume that SQLMACH1 is the default local application server and RMTSERV1 is a remote application server. Connections have been established for SQLMACH1 and online access to RMTSERV1 through the online support is allowed.

```
User:   2 cird ?
System: F2 002 The default server is SQLMACH1.
        F2 002 There are connections to server SQLMACH1.
        F2 002 Online access to remote RMTSERV1 is allowed.
        F2 002 ---------------------------------------------------
```

Figure 50. Example of CIRD with ?

Stopping the Online Support -- The CIRT Transaction

While the online support is enabled, it uses CICS resources (storage) and application server resources (agents). At certain periods of the day, you may want to free these resources and prevent online access to the application server. You may, for example, want to allow only batch access to the application server for purposes of loading a large amount of data. For either of these situations, the operator can disable the online support by entering the CIRT transaction.

To end DB2 Server for VSE online support, issue the CICS CIRT transaction. The syntax of the CIRT transaction is as follows:

```
►►──CIRT──┬─,─────────┬──┬─,─────┬──┬─,─────────┬──────────────────────────►◄
          └─password,─┘  └─mode,─┘  └─interval─┘
```

Figure 51. CIRT Transaction Syntax

Table 13. CIRT Transaction Parameters

Parameter	Default	Description
PASSWORD (positional parameter 1)	SQLDBAPW	This password establishes the operator's authority to terminate the online access to the application server. It must be the same password that was supplied for the CIRA or CIRB transaction. Refer to "Password Implications on Online Resource Adapter Termination" on page 116 for more details.

Table 13. CIRT Transaction Parameters (continued)

Parameter	Default	Description
MODE (positional parameter 2)	NORMAL	This parameter establishes the shutdown mode: NORMAL or QUICK. When remote application servers are accessed by the online support, CIRT NORMAL will complete only when all conversations to the remote application servers are deallocated. When you specify NORMAL, the CIRT transaction prevents new online users from accessing the application server. Users who are already doing work, however, can finish. When all users complete their work, no online users can use the application server. When you specify QUICK, online access to local application servers is ended immediately. Online users accessing a local application server cannot finish their work. Their current logical units of work are rolled back (unless they are already processing a COMMIT WORK). You can change from NORMAL to QUICK. However, once the MODE is QUICK, you cannot change it back to NORMAL. When remote application servers are accessed by the online support and you specify QUICK, online access to the remote application server is *not* ended immediately. Online users accessing a remote server can finish their unit of work, but cannot start a new logical unit of work. QUICK mode is not supported for a remote application server.
INTERVAL (positional parameter 3)	30 (seconds)	The number of seconds that the CIRT transaction should delay before freeing the terminal. The value must be an integer value between 0 and 3600. This parameter controls the availability of the CICS terminal (or operator console) once you issue the CIRT transaction. The CICS terminal (or VSE operator console) used to activate the CIRT transaction is unavailable until the transaction ends. This could be a long time if the online application is long-running or if a user left without correctly ending the terminal session. If you issue CIRT PASSWORD,NORMAL the terminal is not available until all online DB2 Server for VSE users complete their work. Even with CIRT PASSWORD, QUICK there may be some delay before the CICS terminal allows the CIRT terminal to complete its cleanup process. The value you specify here represents an interval of time measured in seconds. If the CIRT transaction does not finish immediately, it waits the amount of time you specify. When this time ends, the CIRT transaction tries once again to finish processing. If the CIRT transaction does not finish successfully, you receive a message telling you to retry the CIRT transaction later. After issuing the message, the CIRT transaction ends. The shutdown mode is still in effect (the specified DB2 Server for VSE system is in the process of shutting down), and the terminal is available for your use.

If links or online access to multiple application servers exist, they will all be removed. Once all of the links and/or online access have been removed, the online resource adapter is terminated.

The following examples assume that SQLVM, SQLMACH1 and SQLMACH2 are local application servers, and SQLMACH8 is a remote application server.

```
msg f2
AR 015 1I40I READY
2 cirt
F2-002 ARI0455I Connections to SQLVM are disabled.
F2-002 ARI0455I Connections to SQLMACH2 are disabled.
F2-002 ARI0455I Connections to SQLMACH1 are disabled.
F2-002 ARI0455I Online access to SQLMACH8 is disabled.
F2-002 ARI0413I Resource Adapter ARI0OLRM is disabled.
```

Figure 52. Example of CIRT with Connections to Four Applications Servers

Note that the message `ARI0413I Resource Adapter ARI0OLRM is disabled` is not displayed until the last application server connections and APPC conversations have been severed.

When the online resource adapter is not active, the CIRA and CIRR transactions are invalid. The online resource adapter needs to be enabled with the CIRB transaction before the CIRA and CIRR transactions can be used.

```
F2-002 ARI0413I Resource Adapter ARI0OLRM is disabled.
2 cira ,,,sqlmach1
F2-002 ARI0411I Resource Adapter is not enabled.
2 cirr ,,,sqlmach1
F2-002 ARI0411I Resource Adapter is not enabled.
```

Figure 53. Example of CIRA and CIRR after CIRT

Effect of a Shutdown on Online Applications

In the NORMAL mode, CIRT prevents new LUWs from being started. As LUWs end, the links to the local application server are disconnected and APPC conversations to the remote application server are deallocated. (The NORMAL process allows for the normal end of all online LUWs.) After all links are disconnected and all APPC conversations are deallocated, the CICS storage resources are freed, and application access to the DB2 Server for VSE online support is no longer allowed.

In the QUICK mode, links to the local application server are immediately disconnected. Some online LUWs may be interrupted. The CICS storage resources are freed, and application access to online support is no longer allowed.

With QUICK, when the links are disconnected, the application server partition is posted by the operating system. The post causes the database manager to do an internal ROLLBACK WORK for all LUWs that were not committed or at a synchronization point (that is, those LUWs that were prepared for COMMIT or ROLLBACK).

While the CIRT transaction is ending access in QUICK mode, the CICS transactions that access the application server can be ended by CICS with an abend code of AEY9, ASP7, or ASRA. To allow for normal transaction shutdown, then, you should either use the CIRD transaction to determine which transactions accessing the application server are still active and wait until they are complete, or use the CIRT transaction with the NORMAL option which allows all active users to finish their work.

The QUICK mode is not supported when you are ending online access to a remote server. In this case, the QUICK mode is changed to NORMAL mode.

Terminal Availability During Online Shutdown

The terminal used to activate the CIRT transaction for NORMAL or QUICK is unavailable until the transaction ends. This could be a long time for a large online application or for an online application controlled by a CICS terminal operator who is not at the console. There are two conditions when CIRT may need to wait (in CICS terms, delay for an interval of time):

- In the NORMAL mode, the process must wait until all LUWs complete normally.
- In both NORMAL and QUICK modes, after all connections and APPC conversations to the application server are severed, the process attempts to disable itself. The attempt can fail if CICS finds some online transaction that is still active and had access to the application server before CIRT was issued. In this situation, the CIRT transaction cannot complete its clean-up process until that transaction ends.

In the situations described above, CIRT will wait for an interval of time before attempting to complete the cleanup process again. (The default interval of time is 30 seconds. The interval can be specified as an input parameter to CIRT.)

After the delay, the CIRT transaction determines if the condition that caused the wait has passed. If it has, the process completes, and the online support is disabled. If not, CIRT exits by returning to CICS (the shutdown mode is still active and the terminal is free), and message ARI0414I is displayed, prompting the operator to retry the CIRT transaction later.

The operator can proceed in a number of ways to disable the online support:

- The installation may have a policy that work can continue until 5:30 PM. The operator routinely issues `CIRT SQLDBAPW,NORMAL` at 5 PM. Doing this prevents new work from starting. The operator then waits until 5:30 PM and reissue the CIRT transaction to proceed with normal transaction shutdown.
- The operator can use the CICS message transaction CMSG to route messages to selected terminals or users, and CICS CEMT, CIRD or CSMT commands to determine who or what applications are active, or to end the application. After such operator intervention is completed, the CIRT transaction is re-entered and the online support becomes disabled.

  This intervention presupposes that the operator has information about those CICS transactions that access the application server. You may find it useful to keep a list or use a naming convention for all such transactions.
- If the NORMAL process was attempted and could not finish, the operator can escalate the shutdown mode (escalate in the sense that the database manager goes from NORMAL mode, which allows all LUWs to end, to QUICK mode, which immediately stops all access to the application server). To escalate, the operator enters `CIRT SQLDBAPW,QUICK`.

Shutdown Impact to Online Applications

After the online support has been disabled, or before it has been enabled, CICS abnormally ends any transaction that attempts online access to the application server by abending the transaction with abend code AEY9. If an attempt is made to execute a transaction while the online support has not been enabled, the transaction also abends with an abend code AEY9. If an application attempts to use CICS HLPI to access either a CICS/VSE subsystem or non-CICS/VSE subsystem that has not been enabled, the CICS terminal operator receives the CICS/VSE abend code AEY9.

When the shutdown process is active, the following occurs:

- For NORMAL mode, the result depends on the state of the application program. If it is in work, the process has no effect. If the application program is not in work, the online support returns an SQLCODE of -937. A later request by such a program will cause CICS to abnormally end the transaction with the AEY9 abend code.
- For QUICK mode, all initial requests result in the -937 SQLCODE, and a later request will result in the AEY9 abend code.

  Also, for the QUICK mode, the online support cannot participate in the CICS two-phase syncpoint protocol. (For information on this protocol, review the discussion on online application recovery in the *DB2 Server for VSE & VM Database Administration* manual.) When the online support reports to CICS that it is disabling, the result is an ASP7 abend. This is the general abend code that the CICS syncpoint manager uses when a CICS or non-CICS/VSE subsystem cannot participate in the two-phase syncpoint protocol. Online application programs do not regain control for clean-up routines when an ASP7 abend occurs. The ISQL transaction must be ended by the operator with the CICS CSMT or CEMT command.

Password Implications on Online Resource Adapter Termination

The password used on the CIRR and CIRT transactions must be the same one that was used on the CIRA and/or the CIRB transactions. CIRR and CIRT will only shut down the connections to servers where the password matches. If the passwords do not match, that server is not shut down.

Consider the following example:

1. The online resource adapter is started with the command:

   CIRB pw1,5,,,,(SQLMACH1,SQLMACH2)
2. Connections to two new servers are added with the command:

   CIRA ,,,(SQLMACH3,SQLMACH4)
3. Another connection is added to a fifth server with the command:

   CIRA pw2,1,,SQLMACH5

It is not possible to end the online resource adapter with one command in this scenario. The CIRT or CIRR transactions must be run at least three times before the online resource adapter is completely shutdown because three different passwords were used to start it up.

The CIRT transaction issued with no parameters would only shut down the connections to SQLMACH3 and SQLMACH4 because they were the only servers that were started with the default password.

To shut down SQLMACH5, you would have to enter the following command:

CIRT pw2

To bring down the remaining servers and stop the online resource adapter you need to enter:

CIRT pw1 followed by CIRT

The CIRR transaction can also be used, but the server names must be specified. The following shows the CIRR commands that would be equivalent to the CIRT commands in this scenario.

CIRT pw1 is equivalent to CIRR pw1,,,(SQLMACH1,SQLMACH2)

CIRT is equivalent to CIRR ,,,(SQLMACH3,SQLMACH4)

CIRT pw2 is equivalent to CIRR pw2,,,SQLMACH5

If the command:

CIRR ,,,(SQLMACH1,SQLMACH2,SQLMACH3,SQLMACH4,SQLMACH5)

were entered only SQLMACH3 and SQLMACH4 would be disconnected.

Message ARI0464E will be issued for servers SQLMACH1, SQLMACH2 and SQLMACH5 because the passwords do not match.

Similarly, if the command:

CIRR pw1,,,(SQLMACH1,SQLMACH2,SQLMACH3,SQLMACH4,SQLMACH5)

were entered only SQLMACH1 and SQLMACH2 would be disconnected.

Message ARI0464E will be issued for servers SQLMACH3, SQLMACH4 and SQLMACH5 because the passwords don't match.

Chapter 6. Maintaining Database Security

Database security is maintained through the use of authorities and privileges. Only users granted the CONNECT authority for an application server can access it.

For information about DB2 Server for VSE authorities and privileges, refer to the *DB2 Server for VSE & VM Database Administration* manual. This chapter discusses the following topics:

- Protecting VSAM data sets from unauthorized access
- VSAM commands that must not be used against any database
- Controlling access to ISQL
- Access Control to Remote Users

Protecting VSAM Data Sets

You can optionally assign a VSAM MASTERPW- or CONTROLPW-level password to protect all the VSAM data sets that make up the database. If you do, you must use it:

- In every DEFINE statement for the VSAM data sets that make up the database
- When you install the database manager
- When you add new data sets.

Table 15 on page 134 shows how a CONTROLPW is to be coded. Coding for a MASTERPW is identical, except for the keyword.

Each time the application server is started, the DBPSWD=*password* initialization parameter must be supplied. The database manager uses the password when it opens the VSAM data sets. If the password supplied does not match the one defined for the data sets, the operator is prompted to supply the correct one. (See "DBPSWD" on page 50)

VSAM Restrictions

Storage for the database manager is defined by VSAM. However, VSAM does not manage this storage. The VSAM commands such as EXPORT, IMPORT, REPRO, and VERIFY should **never** be used against the database. If an error message is received indicating an OPEN error (RC=74), ignore it and do **not** run VERIFY.

Controlling Access by ISQL Users

In VSE, ISQL is made up of two transactions: ISQL and CISQ. The former controls the CICS terminal, and the latter controls access to the application server. By creating the second transaction dynamically (instead of hard-coding it as CISQ) you can put different departments or different groups of users into different CICS classes. Each group would have different transaction identifiers for both transactions of ISQL. Because the different groups have different CICS classes, you can limit the number of active ISQL users in each group.

To implement this, create any transaction ID for the first transaction. Then, instead of making CISQ the second transaction ID, make it identical to the first one except for the last character, which should be a 2. For example, if there are five departments, you could have chosen these transaction IDs:

```
First            Second
Transaction ID   Transaction ID            Department
--------------   ---------------------     ----------
ISQL             ISQ2                      202
ACCT             ACC2                      ACCOUNTING
SAL              SA2                       SALES
IN               I2                        INVENTORY
P                P2                        PLANNING
```

These examples show how the format works for different identifier lengths. Note that when the first transaction ID is one character (P), the 2 is added (P2). Also note that the first transaction ID cannot end with a 2.

Next, decide what the maximum number of ISQL users for each department should be:

```
First            Second                              Maximum
Transaction ID   Transaction ID    Department   ISQL Users
--------------   --------------    ----------   ----------
ISQL             ISQ2              202          2
ACCT             ACC2              ACCOUNTING   3
SAL              SA2               SALES        4
IN               I2                INVENTORY    3
P                P2                PLANNING     2
```

Next, specify the CICS parameters TRANSID, TCLASS, and CMXT as follows:

- TRANSACTION parameter in the CICS System Definition File

  You must code an entry for each transaction ID defined. In the above example these are: ISQL, ISQ2, ACCT, ACC2, SAL, SA2, IN, I2, P, and P2. The TRANSACTION must specify the particular transaction ID (for example, TRANSID=ISQ2 for the ISQ2 transaction), and the program name parameter should reference the same program as CISQ or ISQL.
- TCLASS parameter and CMXT parameter in the DFHSIT

  To fully understand these two parameters, it is best to consider them together. To implement the above example, you would code them as follows:

```
DEFINE TRANSACTION(ISQL) GROUP(DB2710) PROGRAM(ARIITRM)          *
    TWASIZE(300) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES)

DEFINE TRANSACTION(ISQ2) GROUP(DB2710) PROGRAM(ARIISQL)          *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES) TCLASS(1)

DEFINE TRANSACTION(ACCT) GROUP(DB2710) PROGRAM(ARIITRM)          *
    TWASIZE(300) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES)

DEFINE TRANSACTION(ACC2) GROUP(DB2710) PROGRAM(ARIISQL)          *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES) TCLASSS(2)

DEFINE TRANSACTION(SAL) GROUP(DB2710) PROGRAM(ARIITRM)           *
    TWASIZE(300) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES)

DEFINE TRANSACTION(SA2) GROUP(DB2710) PROGRAM(ARIISQL)           *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES) TCLASS(3)

DEFINE TRANSACTION(IN) GROUP(DB2710) PROGRAM(ARIITRM)            *
    TWASIZE(300) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES)

DEFINE TRANSACTION(I2) GROUP(DB2710) PROGRAM(ARIISQL)            *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES) TCLASS(4)

DEFINE TRANSACTION(P) GROUP(DB2710) PROGRAM(ARIITRM)             *
```

```
     TWASIZE(300) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES)

 DEFINE TRANSACTION(P2) GROUP(DB2710) PROGRAM(ARIISQL)           *
     TWASIZE(0) INDOUBT(BACKOUT) SPURGE(NO) TPURGE(YES) TCLASS(5)
```

These TCLASS values correspond to the positional values in the CMXT parameter. These values are arbitrary, but you should set them up so that a transaction’s TCLASS value corresponds to its CMXT positional parameter. In the above example, ISQ2 has a TCLASS value of 1. This means that it is in class 1, which corresponds to the first positional value on the CMXT parameter. The first positional parameter value for CMXT is 2. This means that the maximum number of transactions that can be active in class 1 (TCLASS=1) is 2. Therefore, the number of active Department 202 users of the ISQL-ISQ2 transactions is limited to 2. The same is true for the other TCLASS and CMXT positional values. (For unspecified CMXT values, the default is 1.)

Controlling Access by Remote Users

The AXE transactions must be installed in one or more groups with appropriate security mechanism provided by CICS or by an external security manager. Local CICS users should not be able to start any of these transactions.

The security levels are:

- Bind-time Security

  This is the session level security that can be used if the partner LU supports LU-to-LU verification.
- Transaction Security

  This controls the link authorization to attach the AXE transaction.

For more information on CICS security, see the *VTAM Resource Definition Reference* and the *CICS/VSE Intercommunication Guide* manuals.

After deciding on the security level, specify the TCLASS parameter in the CICS System Definition for the AXE transaction. An example is shown below.

```
DFHSIT    ...,CMXT=(5,6,2,1,,),...

DEFINE TRANSACTION(AXE1) GROUP(DB2710) PROGRAM(ARICAXED)       *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(YES) TPURGE(YES) TCLASS(1)
DEFINE TRANSACTION(AXE2) GROUP(DB2710) PROGRAM(ARICAXED)       *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(YES) TPURGE(YES) TCLASS(2)
DEFINE TRANSACTION(AXE3) GROUP(DB2710) PROGRAM(ARICAXED)       *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(YES) TPURGE(YES) TCLASS(3)
DEFINE TRANSACTION(AXE4) GROUP(DB2710) PROGRAM(ARICAXED)       *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(YES) TPURGE(YES) TCLASS(4)
DEFINE TRANSACTION(AXE5) GROUP(DB2710) PROGRAM(ARICAXED)       *
    TWASIZE(0) INDOUBT(BACKOUT) SPURGE(YES) TPURGE(YES) TCLASS(5)
```

Note: There are a maximum of ten classes which can be defined to CICS. Ensure that when you design your security methodology, you do not commit a class for more than one purpose. That is, if you use TCLASS 1 for limiting ISQL users, then that is the same limit if you assign TCLASS 1 for limiting remote users.

In the example, 5 remote users can access the application server with TPN AXE1, 6 remote users can access the application server with TPN AXE2 and so on. If AXE4 is a privileged TPN for a critical application, only that application is given access to TPN AXE4. This way, the application has exclusive use of a real agent on the application server.

DRDA Security

If any packages need to be secure (not viewed by unauthorized persons), the application can be preprocessed into a private bind file that is only accessible to the authorized person(s). VSAM security control mechanism could then be used to protect this private VSAM file so that unauthorized access is denied. You can change the JCL of the preprocessor to specify the *file_id* of the private VSAM Bind file on the SQLBIND DLBL statement and use this *file_id* with the CBND transaction.

With CICS/VSE, you can only establish SECURITY=SAME conversions with remote partners. Therefore, DRDA security checking is performed during handshaking.

Enabling Password Encryption and Decryption for DRDA

The DB2 Server and requester for VSE can decrypt and encrypt a password, respectively. To enable this service execute job control member ARIS73XD. This job can be executed any time. It will generate two phases, ARICSEC and ARICSCC. ARICSCC is used by the Online Resource Adapter to provide the same support. The server support is activated the next time the DB2 Server for VSE database manager is started. Requester support is activated the next time the resource adapter is recycled.

Chapter 7. Managing Database Storage

This chapter discusses:

- Database storage concepts
- Adding dbspaces to a database
- Expanding the page tables in the directory
- Acquiring dbspaces for packages
- Managing storage pools.

Storage Concepts

A database contains user data objects (tables and indexes), and supporting information maintained by the database manager. Specifically, it contains:

- A *directory* which is a data set containing database control information, including mappings of the dbspaces to their addresses on DASD. The directory relates the logical database image to the physical storage used.
- Either one, two, or four *log data sets* which hold records that describe each change made to the database. If any changes made to the data must be undone or redone, logs can be used to restore the data to a consistent state.
- One or more *storage pools*, which are collections of data sets called database extents (*dbextents*). This is where the actual data is stored.

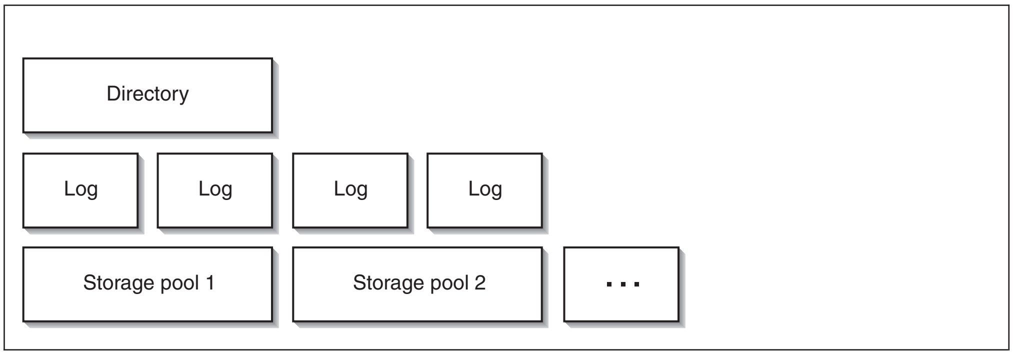


Figure 54. The DB2 Server for VSE Database

A *dbextent* is an allocation of actual DASD space. *Storage pools* are composed of one or more dbextents. The size of a storage pool can be increased by adding more dbextents, or reduced by deleting existing ones. Each dbextent is the primary allocation of a VSAM data set. When dbspaces are assigned to a storage pool and their pages are filled, physical DASD pages are taken from the dbextents of the storage pool.

Storage pools can be defined so that they are either recoverable or nonrecoverable. By default, storage pools are recoverable, that is, the database manager does full recovery for them. For nonrecoverable storage pools, only limited recovery is done. For more information on nonrecoverable storage pools, refer to "Nonrecoverable Storage Pools" on page 177.

A *dbspace* is a logical allocation of space in the database, divided into 4096-byte blocks called pages. A dbspace is not a real allocation of DASD space, but only an allocation of page tables in the directory. These page tables map logical dbspace pages to DASD locations. The database manager dynamically allocates real DASD storage space to support dbspace pages on a demand basis so unused pages do not occupy DASD space.

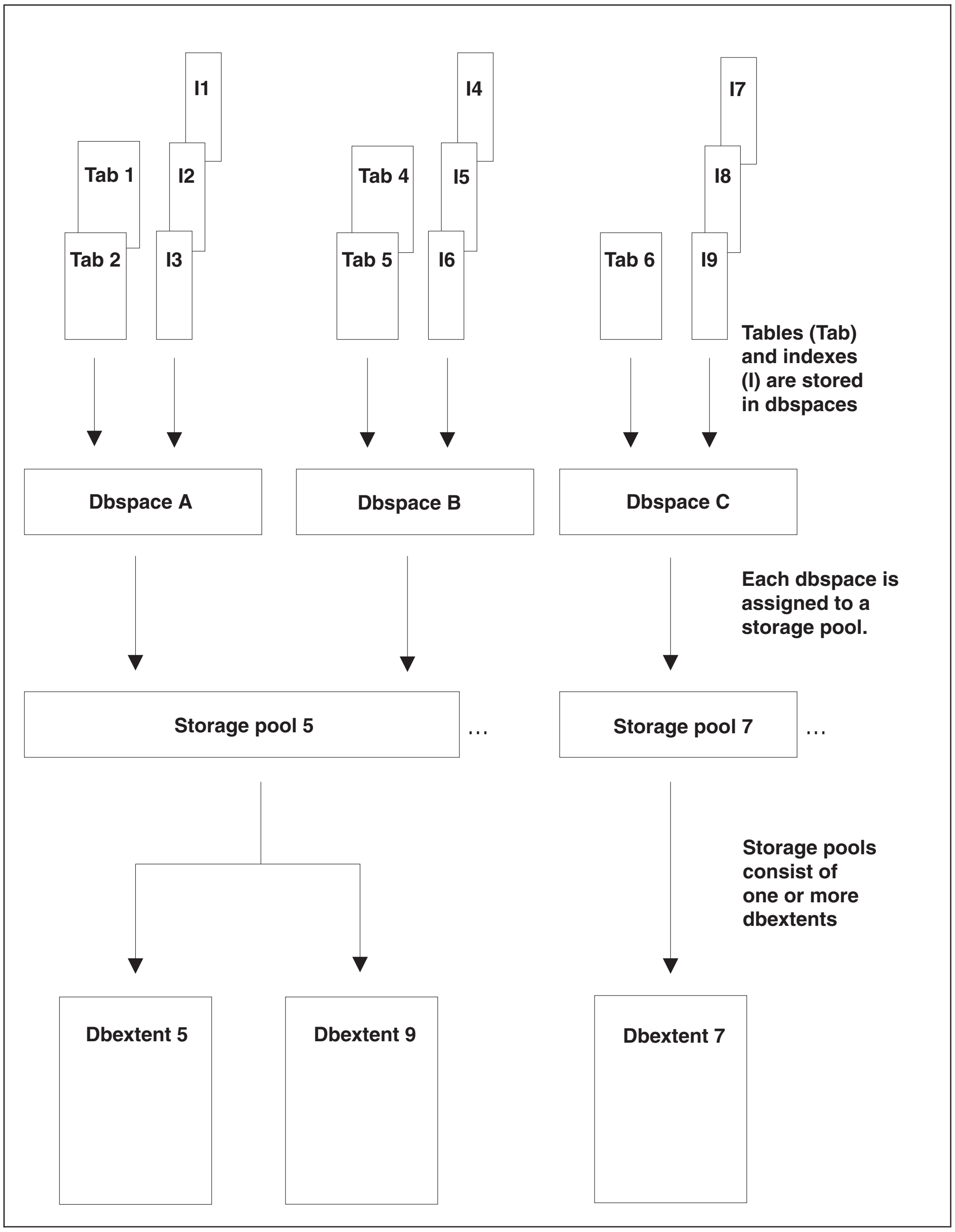

I1
Tab 1
I2
Tab 2
I3
I4
Tab 4
I5
Tab 5
I6
I7
I8
Tab 6
I9
Tables (Tab) and indexes (I) are stored in dbspaces
Dbspace A
Dbspace B
Dbspace C
Each dbspace is assigned to a storage pool.
Storage pool 5
...
Storage pool 7
...
Storage pools consist of one or more dbextents
Dbextent 5
Dbextent 9
Dbextent 7

Figure 55. Physical Database Concepts

How Information is Stored in Dbspaces

Tables and their indexes are stored in dbspaces. At the beginning of every dbspace are one to eight *header pages*, which contain control information on the tables and indexes that follow. Next come data pages, which hold the rows of the tables. At the end are index pages, which hold the index entries. A page in a dbspace is defined as a header page, a data page, or a index page, when the dbspace is acquired. Figure 56 on page 125 shows how information is stored in a dbspace.

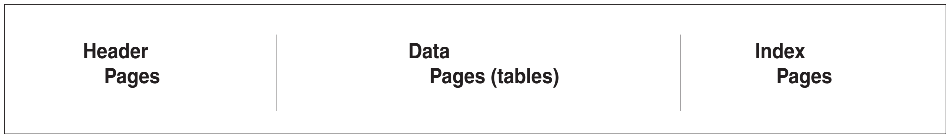


Figure 56. Table and Index Storage in a Dbspace

When a table is created, its creator can either assign it to a dbspace explicitly by specifying a dbspace in the CREATE TABLE statement, or can let the database manager assign it to a default dbspace. Any indexes created on the table obtain their storage from the same dbspace as that table.

Figure 55 shows two tables and their indexes in dbspace A, two tables and their indexes in dbspace B, and one table with three indexes in dbspace C.

The potential capacity of a dbspace is fixed when it is defined with the ADD DBSPACE command. A dbspace can hold up to 255 tables along with their indexes.

More than one table can be stored in the same dbspace, but a table cannot reside in multiple dbspaces. If you store multiple tables in a dbspace, be aware that the database manager may store rows from different tables on the same data pages. For performance reasons, it is frequently desirable to have only one table per dbspace. (Index entries from different indexes are never stored on the same page.)

There are three types of dbspaces: private, public, and internal. For private data, there should be one *private* dbspace reserved for each user. These are locked at the dbspace level, so the database manager does not incur unnecessary overhead while users are accessing their own private data. Any tables that are to be accessed by multiple users who will be doing UPDATE, INSERT, or DELETE operations should be placed in *public* dbspaces, which have page- or row-level locking to support concurrent access. *Internal* dbspaces are temporary spaces used only by the database manager to perform tasks such as sorting.

Adding Dbspaces to the Database

Before tables and indexes can be stored in a dbspace, the dbspace must be *added*, and then *acquired*. Adding a dbspace to a database consists of reserving page tables in the directory, assigning the dbspace to a storage pool, and specifying it as public or private.

The ADD DBSPACE Operation

To create new dbspaces, use the ADD DBSPACE operation. The application server must be running in single user mode (SYSMODE=S), with STARTUP=S.

Specify each dbspace to be added on a SYSIPT input record that contains the type (public or private), the size (number of pages), and, optionally, the storage pool assignment. (The default storage pool number is 1.) The number you specify for the size should be a multiple of 128, since directory page tables are allocated in multiples of 128-page table entries. If it is not, the database manager rounds it up to the next higher multiple of 128. Separate all parameter values by at least one blank. Figure 57 on page 126 shows an example.

```
// JOB ADD DBSPACES
// EXEC PROC=ARIS75SL
// EXEC PROC=ARIS75DB
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='DBNAME=SQL_DB1,SYSMODE=S,STARTUP=S'
   PUBLIC 1024 7
   PUBLIC 1024 8
   PRIVATE 256 5
   PRIVATE 256 5
   PRIVATE 256 5
   PRIVATE 256 5
   INTERNAL 50 1024 9
/*
/&
```

Figure 57. Sample ADD DBSPACE Control Statements

On the last dbspace specification record you must specify the internal dbspaces to be defined. This record contains the keyword INTERNAL, the number of internal dbspaces to be supported, the size of each (in number of pages), and, optionally, the storage pool assignments. Internal dbspaces can be assigned to either recoverable or nonrecoverable storage pools. However, for performance reasons, the internal dbspaces should not be assigned to storage pool 1 and preferably should be stored in their own storage pool. Internal dbspaces can also be stored in a virtual disk. For more information on the performance benefits of virtual disk support, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

It is necessary that you respecify the internal dbspace values each time you add a new public or private dbspace, even if you are not changing these values from what they were before. The internal dbspace specification overrides the previous one, including changing the storage pool assignment.

Note: You may sometimes want to change the internal dbspace specifications for reasons other than adding new user dbspaces. To do this, simply run the ADD DBSPACE operation omitting the control statements for public or private dbspaces, and enter the number of internal dbspaces you want.

Considerations for Adding Dbspaces

The ADD DBSPACE operation updates the directory and the catalog tables in the database. Only the updates to the catalog tables are recorded in the log; updates to the directory are not. Because of this, you can have a problem if you normally archive the database, and then try to restore it. Suppose the following events occur:

1. You do a database archive.
2. Later, you add dbspaces.
3. Later, users acquire and use those dbspaces.
4. You do an archive restore using the archive file that you created in step 1 and, if you use LOGMODE=L, the subsequent log archives.

The directory and the database are not synchronized. The directory has been restored from a database archive file that does not reflect the ADD DBSPACE operation. The database is also restored from that file; but its restore includes the updates recorded in the log or log archives, which do reflect the ADD DBSPACE operation. Thus, the directory does not include the new dbspaces but the database does.

To prevent this problem, archive the database immediately after the ADD DBSPACE operation, as follows:

1. After you add the dbspaces, warm-start the application server in multiple user mode (SYSMODE=M) with LOGMODE set to L or A.
2. Immediately take a new database archive, with either the ARCHIVE, SQLEND ARCHIVE, or SQLEND UARCHIVE command. (If you use SQLEND UARCHIVE, remember to take the user archive after the application server ends.)

Following this procedure will ensure that your current database archive reflects the added dbspaces. (See “Archiving Procedures” on page 152 and “Restoring the Database” on page 158 for more information on archiving and restoring procedures.)

If you do log archiving and restore the database using a database archive taken before the ADD DBSPACE operation, the same problem that was described above occurs. If you use a back-level database archive and subsequent log archives to restore the database, the database archive that records the addition of the dbspaces is skipped: the directory is restored from the back-level database archive and does not show the addition of the dbspaces, but the subsequent log archives do.

If you used the ADD DBSPACE operation only to reconfigure your internal dbspaces, restoring a back-level database does not unsynchronize the directory and database, since information about internal dbspaces is stored in the directory but their use is not recorded in the database. Thus, if you restore a back-level database, the number and size of the internal dbspaces return to the back-level values.

The ADD DBSPACE operation is a two-phase process. The first phase updates the database directory with the information about the new dbspace. The second updates the SYSTEM.SYSDBSPACES catalog table.

Completion of the first phase is indicated by the message:

```
    ARI0915I  DBSPACE ADDED TO DATABASE
```

If an abnormal end occurs before message ARI0915I is issued, restart the ADD DBSPACE operation from the beginning. If an abnormal end occurs after message ARI0915I is issued, restart the ADD DBSPACE operation by doing a start up of the application server as follows:

```
// EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,STARTUP=W,PROGNAME=ARISEGB'
```

Initialization Parameters for ADD DBSPACE

Table 14 on page 128 shows the initialization parameters that you can specify for the ADD DBSPACE operation.

Table 14. Initialization Parameters for the ADD DBSPACE Operation

Parameter	Default	Minimum	Maximum
DBNAME=name	SQLDS	---	---
SYSMODE=S	---	---	---
PARMID=name	None	---	---
STARTUP=S	---	---	---
DBPSWD=password	None	---	---
LOGMODE=Y|A|L|N	Y	---	---
DSPLYDEV=L|C|B	L	---	---
DUMPTYPE=P|F|N	P	---	---
TRACDBSS=*nnn...*	Zeros	Zeros	Twos
TRACRDS=*nnnnnn*	Zeros	Zeros	Twos
TRACDSC=*nnnnnn*	Zeros	Zeros	Twos
TRACCONV=*n*	Zero	Zero	Twos
TRACSTG=*n*	Zero	Zero	Ones
TRACEBUF=*n*	Zero	Zero	99999

The DBNAME, SYSMODE and STARTUP parameters are required as shown to define the run of the database manager as an ADD DBSPACE operation. In addition, DBPSWD is required if the database VSAM data sets are password protected.

If you have been accessing the database with LOGMODE=A or L, you **must** continue to do so for the ADD DBSPACE operation.

You can use PARMID to specify a source member that contains parameter specifications for the ADD DBSPACE operation.

You can also specify the DSPLYDEV, DUMPTYPE, TRACDBSS, TRACDSC, TRACRDS, TRACCONV, TRACSTG and TRACEBUF parameters. For more information, see “Multiple User Mode Initialization Parameters” on page 47. Because the ADD DBSPACE operation requires that the database manager be run in single user mode, the TRACDBSS and TRACRDS initialization parameters are the only means of doing a trace of the ADD DBSPACE operation. (Operator TRACE commands are only valid when the database manager runs in multiple user mode).

Expanding the Database Directory

When a database is initially generated, a calculation is made to determine which portion of the directory will be set aside for the page map table, and which portion will be used for the allocation bitmaps. The size of the page map table determines the maximum number of DBSPACE pages, that is, the maximum logical size of the database. The size of the allocation bitmap determines the maximum number of dbextent pages, that is, the maximum physical size of the database. As the database grows in size with use, it may run short on either logical or physical space. If it is short on logical space, the ADD DBSPACE operation may fail. If it is short on physical space, the ADD DBEXTENT operation may fail. You can expand the directory to correct these situations.

You can use ARIMEXBD to increase:

- The maximum number of dbspace pages, by expanding the page map table using EXPAND=DBSPACE. See Figure 58.
- The maximum number of dbspace pages and dbextent pages, by expanding the page map table and allocation bitmaps concurrently using EXPAND=ALL. See Figure 58.

Expand the directory as follows:

1. Use the VSAM IDCAMS DEFINE commands to define the VSAM data set for the new directory.

   For information about these commands, see the *Using VSE/VSAM Commands and Macros* manual.
2. Run the utility program ARIMEXBD. An example is shown in Figure 58.
3. Update the DLBL statement for the BDISK to reflect the change.

```
// JOB EXPAND BDISK
// LIBDEF PROC,SEARCH=(PRD2.DB2730)
// EXEC PROC=ARIS75PL     *-- PRODUCTION LIBRARY ID PROC
// EXEC PROC=ARIS75DB     *-- DATABASE ID PROC
// DLBL BDSKNEW,'SQL.BDSKNEW.STARTER.DB',,VSAM                        1
// EXEC ARIMEXBD,SIZE=AUTO,PARM='DBPSWD=password,EXPAND=DBSPACE|ALL' 2
/*
/&
```

Figure 58. Sample ARIMEXBD Job

Notes for Figure 58:

1 In the DLBL statements, the file name for the new directory data set must be BDSKNEW.

2 If the VSAM data sets are password-protected, enter the correct password on the DBPSWD parameter.

Acquiring Dbspaces for Packages

The process of adding a dbspace merely reserves pages for it in the directory. Before it can actually be used, it must be *acquired*. For details of how to acquire dbspaces, see the *DB2 Server for VSE & VM Database Administration* manual.

Packages and view definitions are stored in system dbspaces named SYS0002, SYS0003, ... SYS*nnnn*. Allocation of the initial system dbspace (SYS0002) is performed during database generation. You should probably acquire an additional package dbspace after installation, and then more as needs arise. Because unused dbspaces only require minimal directory space and no data pages, acquiring them is not costly. Thus, if your installation has many packages and views, it is a good idea to acquire several dbspaces for packages in advance for later use.

The database manager stores packages and view definitions as tables. A dbspace can contain up to 255 tables, and can therefore have up to 255 packages and view definitions.

Although packages and view definitions are stored as tables, information about them is found not in the SYSTEM.SYSCATALOG catalog table, but in the SYSTEM.SYSACCESS catalog table. When a dbspace is acquired for packages, 255 empty tables are preallocated in it. For each table that is created, a row is added to the SYSTEM.SYSACCESS catalog table that identifies the package table as unused. Unused package tables can be either available or unavailable. The TNAME value in SYSACCESS for unused package tables is represented either as !0*x* AVAILABLE or ¢0*x* UNAVAILABLE. (The *x* is a number from 1 to 5, which is used internally.) Initially, all package tables in a newly acquired dbspace are unused and available. As packages are created and views are defined, the TNAME value is changed to indicate the package or view name.

As mentioned above, you can usually fit 255 packages in a dbspace. However, if large packages are created, the dbspace pages may fill before all 255 package tables are used. In this situation, all remaining package tables are unused and unavailable and their TNAME value is marked in the dbspace as ¢0*x* UNAVAILABLE. When the dbspace is full, the FREEPCT column of the SYSTEM.SYSDBSPACES catalog table is updated. A FREEPCT of 1 means that space is still available, while a FREEPCT of 0 means that this dbspace is full.

If a package or view is dropped from a dbspace that is not full, the database manager does not drop the package table from the dbspace. Instead, it deletes all the rows from the table, and marks the table as available in the SYSTEM.SYSACCESS catalog table. The table can then be reused.

If a package or view is dropped from a dbspace that has been marked as full (FREEPCT = 0), FREEPCT is reset to 1. Before these package tables can be reused, however, their TNAMEs in the SYSTEM.SYSACCESS catalog table must be changed to indicate that they are available. This is not done immediately, because if it were, the next time someone tried to create a package, the database manager would reuse the table from the package or view that was just dropped. It would try to place the newly created package in a dbspace that is almost full, and it probably would not fit. Thus, if you have used all the space in your package dbspaces, you should acquire another dbspace rather than try to free space by dropping one or two unused packages. The package tables will be marked available the next time the database manager does preallocation.

Preallocation is done when you acquire a new package dbspace. It is also done when you try to create a view or a new package, and there are no available packages. If the database manager cannot find an available package, it looks in all dbspaces that are not full (FREEPCT=1) for package tables that are marked unavailable, and marks them as available.

A user with DBA authority can acquire a package dbspace by issuing the following SQL statement when the database is running in multiple user mode:

```
ACQUIRE PUBLIC DBSPACE NAMED SYSnnnn (PAGES=xxxx)
```

where

nnnn is the number of the package dbspace. (SYS0002 is the initial dbspace, so the next one will be called SYS0003, the next one, SYS0004, and so on.)

xxxx is the number of pages of address space for the dbspace. The usual value is 2048, but you can set it larger or smaller if your programs have a large or small number of SQL statements in them, or if you are adding many views to the database.

You should specify the PAGES parameter because the default value of 128 is usually too small. You can specify NHEADER or allow it to default to 8. The database manager sets PCTFREE to 1, PCTINDEX to 0, and LOCK to PAGE (page locking). If you try to specify any of these parameters, your settings will be ignored.

If no package tables are available in any package dbspace during preprocessing, SQLCODE -945 is returned, and the DBA must acquire another dbspace for packages.

If sufficient space is not available in the dbspace where the database manager attempts to create the package, it returns SQLCODE -946. The user's response depends on the availability of package tables in other dbspaces. If some are available, the user can try to preprocess the program again. (The database manager does not choose the same dbspace again because it sets FREEPCT=0 when the preprocess fails.) If no package tables are available, another dbspace for packages must be acquired.

To get information about unused packages (available and unavailable), issue the following query:

```
SELECT * FROM SYSTEM.SYSACCESS WHERE TNAME LIKE '%AVAILABLE'
```

To determine which package dbspaces are full because all the space is taken, issue:

```
SELECT * FROM SYSTEM.SYSDBSPACES WHERE DBSPACENAME LIKE 'SYS0%'
```

If the FREEPCT value is 0, there is no free space in the dbspace.

To determine which package dbspaces are full because all 255 tables are occupied, issue:

```
SELECT DBSPACENO, COUNT(*) -
  FROM SYSTEM.SYSACCESS -
  WHERE TNAME NOT LIKE '%AVAILABLE' -
  GROUP BY DBSPACENO
```

Dbspaces with a count of 255 have no available package tables. (For information on the syntax of the ACQUIRE DBSPACE and SELECT statements, see the *DB2 Server for VSE & VM SQL Reference* manual.)

Managing Storage Pools

Typically, you set up your database to be supported by multiple storage pools, so that you can control what data resides on what devices, and can manage physical DASD allocations differently for different data. The following sections discuss uses of storage pools and how to define them.

Design Considerations for Storage Pools

A storage pool consists of a large collection of 4-kilobyte DASD pages, called *slots*, for storing allocated public and private dbspace pages and shadow pages (old copies of dbspace pages that have changed since the last checkpoint). Dbspace pages that are not allocated are not stored. For internal dbspaces, slots are occupied only by nonempty pages of data for active logical units of work.

The placement of dbspace pages in storage pool slots is determined by the database manager; however, you control which pool of slots the dbspace pages are assigned to. This allows you to control device utilization and the use of different DASD allocation schemes for different data.

Estimating Storage Requirements

You may often choose to *undercommit* the actual DASD space available for the dbspaces. Because a dbspace cannot be extended after it is defined, and because it is really only a logical allocation of space, many dbspaces are defined to be much larger than needed. As a result, the actual storage pool slots required are fewer than the dbspace sizes imply. The number of dbextent pages should be defined to support the expected number of dbspace pages that will actually be used.

The undercommitting approach to managing storage pools is particularly useful if the tables involved are expected to grow over time. The sizes of the dbspaces are set based on how large the tables can grow, while the size of the storage pool is defined based on current storage requirements. As the tables grow, you can extend the storage pool by adding dbextents to it.

Undercommitting is also useful for supporting internal dbspaces. It is unlikely that you will ever need all the pages of all of the internal dbspaces at the same time. The number of internal dbspaces defined is based on the most the database manager would need at one time, and the size for each is defined based on the worst possible situation that could occur. (Note that internal dbspaces are all the same size.)

If you want to guarantee space availability, or have more dynamic dbspace storage requirements, you should *overcommit* the DASD space available for dbspaces. For example, you might want to do so to handle the storage requirements for private dbspaces. User requests for more or bigger dbspaces can be relatively frequent. Rather than repeatedly going through an ADD DBEXTENT operation, you could overcommit the storage pool for private dbspaces and handle the user requests through the ADD DBSPACE and ACQUIRE DBSPACE operations. (You may still have to run the ADD DBEXTENT operation, but not as often.) For overcommitting, allocate sufficient slots to handle all dbspace pages plus the potential shadow pages.

Controlling Device and Channel Utilization

Storage pools enable you to control device and channel utilization through one of two basic approaches:

- Separating highly referenced dbspaces

  Two highly active dbspaces can be placed on different devices by assigning them to different storage pools and defining the dbextents of these storage pools on different devices.
- Spreading a highly referenced dbspace across devices

  A single highly active dbspace can be spread across multiple devices by defining its storage pool as small, multiple dbextents, each of which is a VSAM data set defined on a different device.

Controlling Data Location

You can allocate a specific table and all its indexes to a specific device or VSAM data set. To do this, create the table in a dbspace with no other tables, assign that dbspace to its own storage pool, and define the dbextents of that pool as the VSAM data sets on the volume that you want.

Monitoring Storage Pools

Use the `SHOW POOL` command to display physical storage information about each storage pool defined, including:

- The total number of pages in the storage pool

- The number of pages being used
- The percentage of the pages in use
- The number of dbextents defined for that storage pool, in the order in which they were defined (which is also the order in which they will be searched for a free page)
- For each dbextent
  - The total number of pages
  - The number of free pages
- A short-on-storage indicator.

You can issue the `SHOW POOL` command from either the operator console or from ISQL. For more information about it, refer to the *DB2 Server for VSE & VM Operation* manual. To see information about reusable deleted dbextent numbers, use the `SHOW POOL DELETED` command.

Maintaining Storage Pools

To maintain storage pools, you:

- Add storage pools to the database

  You add a storage pool to a database by adding a dbextent to a nonexistent storage pool, using the ADD DBEXTENT process described in "Adding Dbextents to a Storage Pool."
- Add storage to existing storage pool

  If any of your storage pools are short on storage, you can use the ADD DBEXTENT process to increase their size.
- Remove storage from storage pools

  You can use the DELETE DBEXTENT process to release DASD for other uses.
- Move dbextents to another device

Adding Dbextents to a Storage Pool

Dbextents can be added to a nonexistent storage pool (which defines a new storage pool), or to an existing storage pool (which increases the size of the storage pool) using the following two-step process:

1. Define the dbextent VSAM data sets
2. Update the database job control
3. Run ARIS250D procedure to add the dbextents.

These steps are described in more detail below.

Step 1: Define the Dbextent VSAM Data Sets: Run the VSAM IDCAMS program to define the VSAM data sets. This step allocates the DASD space and establishes the size of the dbextent. Table 15 shows an example of a job for defining three dbextents.

Table 15. Example of a Job for Allocating Dbextent Data Sets

```
// DLBL IJSYSCT,'AMASTCAT',,VSAM
// EXEC IDCAMS,SIZE=AUTO
   DEFINE SPACE -
          (DEDICATE -
           VOL(DBDISK7)) -
           CAT(SQLCAT01/PASSWORD)
   DEFINE CLUSTER -
          (NAME(SQL.DDSK15.DBNAME01.DB) -
           CNVSZ(4096) -
           CYL(50) -
           NONINDEXED -
           VOL(DBDISK7) -
           CONTROLPW(PASSWORD) -
           RECSZ(4089) -
           REUSE -
           SHR(1)) -
           CAT(SQLCAT01/PASSWORD)
   DEFINE SPACE -
          (DEDICATE -
           VOL(DBDISK8)) -
           CAT(SQLCAT01/PASSWORD)
   DEFINE CLUSTER -
          (NAME(SQL.DDSK16.DBNAME01.DB) -
           CNVSZ(4096) -
           CYL(20) -
           NONINDEXED -
           VOL(DBDISK8) -
           CONTROLPW(PASSWORD) -
           RECSZ(4089) -
           REUSE -
           SHR(1)) -
           CAT(SQLCAT01/PASSWORD)
   DEFINE CLUSTER -
          (NAME(SQL.DDSK17.DBNAME01.DB) -
           CNVSZ(4096) -
           CYL(30) -
           NONINDEXED -
           VOL(DBDISK8) -
           CONTROLPW(PASSWORD) -
           RECSZ(4089) -
           REUSE -
           SHR(1)) -
           CAT(SQLCAT01/PASSWORD)
/*
```

Note: For minimum space allocation values, see Table 41 on page 342.

In this example, one dbextent data set called SQL.DDSK15.DBNAME01.DB is defined on volume DBDISK7, and two more, SQL.DDSK16.DBNAME01.DB and SQL.DDSK17.DBNAME01.DB, are defined on volume DBDISK8.

You can move dbextents between device types as long as the dbextent is not larger than the size of the device. When you define dbextents, you should keep this in mind. For example, if you defined a single dbextent of 600000 blocks on a 9335 device, you could not move that dbextent to a 9332 device which is limited to 360032 blocks. However, if you defined three dbextents, each of 200000 blocks, on a 9335 (for a total of 600000 blocks), you could move them to three 9332 devices.

Step 2: Update the Database Job Control: If you are using cataloged procedures to include the DLBL statements for your database, you must update those procedures to include the DLBL statements for the new dbextents. For the example shown in Table 15, you would add the following three DLBL statements:

```
// DLBL DDSK15,'SQL.DDSK15.DBNAME01.DB',,VSAM
// DLBL DDSK16,'SQL.DDSK16.DBNAME01.DB',,VSAM
// DLBL DDSK17,'SQL.DDSK17.DBNAME01.DB',,VSAM
```

Step 3: Run the ADD DBEXTENT operation: Run the ARIS250D procedure to add dbextents to the storage pool. This step updates the database directory to include the control information for the dbextents. If the dbextents are being added to a new storage pool, this procedure also defines the new storage pool as are being recoverable or nonrecoverable. Multiple dbextents can be defined in one run of each of these jobs. For a description of this procedure, see "Using the ARIS250D Procedure" on page 136.

Deleting Dbextents from a Storage Pool

Deleting a dbextent does not delete any data in the database. Data in the deleted dbextent is first moved to another dbextent in the same pool before it is removed from the database.

Dbextents can be deleted from a storage pool using the following three-step process:

1. Run ARIS250D procedure to delete the dbextents
2. Update the database job control
3. Delete the dbextent VSAM data set.

These three steps are described in more detail below.

Step 1: Run the DELETE DBEXTENT operation: Run the ARIS250D procedure to delete dbextents from the storage pool. This step updates the database directory to remove the control information for the dbextents. For a description of this procedure, see "Using the ARIS250D Procedure" on page 136.

Attention

You must not delete the only dbextent from the storage pool that contains the internal dbspaces.

Step 2: Update the Database Job Control: If you are using cataloged procedures to hold the DLBL statements for your database, you must remove them for the deleted VSAM data sets. For the example shown in Figure 59, you would delete the following three DLBL statements from your job control procedures:

```
// DLBL DDSK15,'SQL.DDSK15.DBNAME01.DB',,VSAM
// DLBL DDSK16,'SQL.DDSK16.DBNAME01.DB',,VSAM
// DLBL DDSK17,'SQL.DDSK17.DBNAME01.DB',,VSAM
```

Step 3: Delete the Dbextent VSAM Data Sets: Run the VSAM IDCAMS program to physically delete the DASD space for the dbextent. Figure 59 shows how to delete the three VSAM data sets that were defined in Table 15 on page 134.

```
// JOB DELETE DBEXTENT DATA SET
// DLBL IJSYSCT,'AMASTCAT',,VSAM
// EXEC IDCAMS,SIZE=AUTO
   DELETE (SQL.DDSK15.DBNAME01.DB/PASSWORD)
   DELETE (SQL.DDSK16.DBNAME01.DB/PASSWORD)
   DELETE (SQL.DDSK17.DBNAME01.DB/PASSWORD)
/*
/&
```

Figure 59. Example Job Step for Deleting Dbextent Data Sets

Note: You can move a dbextent from one storage pool to another by deleting it and adding it back to the new pool; however, you cannot delete, add, and then delete the same dbextent in a single run.

Using the ARIS250D Procedure

A dbextent is added to or deleted from the database using the procedure ARIS250D shown in Figure 60. This procedure starts the application server in single user mode (SYSMODE=S) with STARTUP=E. The job control to run this procedure is shown in Figure 61. The specifications for the dbextents to be added or deleted are provided in the member ARISADD, shown in Figure 63 on page 137.

```
      ***********************************************************
      * ARIS250D: ADD AND DELETE DBEXTENTS
      *    THE PROGRAM SCANS THE INPUT TWICE.  FIRST PASS
      *    TO CHECK FOR ERRORS, SECOND PASS TO EXECUTE.
      ***********************************************************
      // EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,STARTUP=E'
      READ MEMBER ARISADD
      /*
      READ MEMBER ARISADD
      /*
      /&
```

Figure 60. Procedure ARIS250D

The job control to run ARIS250D is shown in the figure below:

```
// JOB ARIS75HD ADD AND DELETE DBEXTENTS
// LIBDEF PROC,SEARCH=(PRD2.DB2730)
// EXEC PROC=ARIS75PL     *-- PRODUCTION LIBRARY ID PROC
// EXEC PROC=ARIS75DB     *-- DATABASE ID PROC
// EXEC PROC=ARIS250D     *-- ADD AND DELETE DBEXTENT PROC
/&
```

Figure 61. Example Job Control for ARIS250D procedure

If you are using your own startup job stream instead of ARIS250D, you must code READ MEMBER ARISADD twice, and separate each line with /*. If you are coding the control statements in stream, you must code them twice, and separate them with /* as shown in Figure 62. The ARISQLDS program requires two sets of identical specifications for efficiency reasons. The first set is for syntax checking, and the second set is for processing.

```
// JOB ADD AND DELETE DBEXTENTS
// LIBDEF PROC,SEARCH=(PRD2.DB2730)
// EXEC PROC=ARIS75PL
// EXEC PROC=ARIS75DB
// EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,STARTUP=E'
   POOL   8 NOLOG
   DELETE 3 1
   DELETE 2 2
   ADD    6 8
   DELETE 4
   ARCHIVE
/*
   POOL   8 NOLOG
   DELETE 3 1
   DELETE 2 2
   ADD    6 8
   DELETE 4
   ARCHIVE
/*
/&
```

Figure 62. Example Job Control for Adding or Deleting Dbextents

The following figure contains examples of the control statements typically found in a member such as ARISADD, used by the ARIS250D procedure to add or delete dbextents.

```
POOL   8 NOLOG
DELETE 3 1
DELETE 2 2
ADD    6 8
DELETE 4
ARCHIVE
```

Figure 63. Example ARISADD for Adding or Deleting Dbextents

If the ARCHIVE control statement is specified, it must come last. The valid options are ARCHIVE (database archive), UARCHIVE (user archive) or NOARCHIVE (no archive). If you do not specify it, the default (ARCHIVE) is used.

Attention

The database **cannot** be restored from an archive taken prior to the deletion of a dbextent after the dbextent is removed from the database. Therefore, the user should choose ARCHIVE or UARCHIVE to backup the database.

The optional POOL control statements must precede the statements that define the dbextents. They are required only for defining new nonrecoverable storage pools with POOL(NOLOG). They are unnecessary if you are adding dbextents to an existing pool because a storage pool's status has already been defined as either nonrecoverable or recoverable. POOL statements are also not necessary for new recoverable storage pools, because by default, storage pools are recoverable. The POOL control statement shown in Figure 63 defines storage pool 8 as nonrecoverable.

You cannot specify pool number 1 on any POOL control statement.

Attention

Once a storage pool is defined as either nonrecoverable or recoverable, you must not change it from recoverable to nonrecoverable (or from nonrecoverable to recoverable).

The records following the POOL control statements contain the dbextent definitions. Each control statement must contain a control word (ADD or DELETE) and the specification of one dbextent. The first number in the input record is the number designator of the dbextent. The second number, if specified, is the number designator of its storage pool. (For the ADD action, if this number is not specified, the default is storage pool 1; for the DELETE action, the default is the storage pool where the dbextent resides.) The numbers must be separated by at least one blank.

When you add a dbextent, its number must either be one more than the number of dbextents currently defined, or the number of any dbextent that was deleted by the DELETE DBEXTENT operation. The total amount of space allocated in the directory as the dbextent control area is fixed for a database, and cannot be changed without regenerating the database. When a dbextent is deleted, the control area is not compressed. Therefore, you should reuse deleted dbextent numbers whenever possible so as to reuse the directory control area. Figure 64 shows area the dbextent control in the directory.

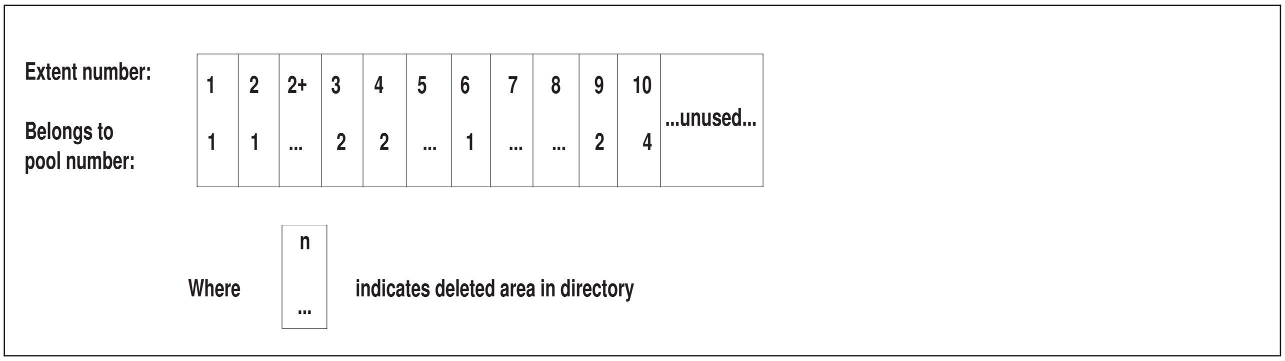


Figure 64. Dbextent Control Area in the Database Directory

In this example, a new dbextent can take on the numbers 5, 7 or 8, which are available for reuse, or 11, which is the next sequential number. The value 2+ indicates that there is empty directory space between dbextents 2 and 3. Because no dbextent number is associated with this space, you must first delete dbextent 2 or dbextent 3 to reclaim it.

You can determine the number of dbextents currently defined in a database by using the `SHOW POOL` operator command. To determine the maximum number of dbextents or storage pools that can be defined for the database, issue the `SHOW DBCONFIG` operator command. For more information, see the *DB2 Server for VSE & VM Operation* manual.

You can determine the deleted dbextent numbers that are available to be reused by using the `SHOW POOL DELETED` command. There is a maximum size associated with each deleted dbextent number. The maximum size is determined by the previous use of the dbextent number. The highest number is an exception; if it is deleted, the control area it used to occupy will be combined with the rest of the free area and this number will be treated as if it has never been used.

For example, if dbextent 10 in Figure 64 on page 138 above is deleted, the control area in the directory will look like Figure 65.

Extent number:	1	2	2+	3	4	5	6	7	8	9	...unused...
Belongs to pool number:	1	1	...	2	2	...	1	...	...	2	

Figure 65. Dbextent Control Area in the Directory after Dbextent 10 Is Deleted

When the `SHOW POOL DELETED` command is issued, dbextent number 10 will not be listed.

Storage pool numbers can range from 1 to MAXPOOLS, where MAXPOOLS is the maximum number of storage pools for the database, as specified during the database generation. Storage pool numbers can be used in any sequence.

Considerations for Adding and Deleting Dbextents: Neither the ADD nor the DELETE DBEXTENT operation is recorded in the log. Because these operations update the directory, and not the database itself, you can encounter a problem if you normally archive the database, and then try to restore it. For an ADD DBEXTENT operation, suppose the following events occur in the following order:
1. You do a database archive
2. You add dbextents
3. Users use data from those dbextents
4. You do an archive restore using the archive file from number 1 above and, if you use LOGMODE=L, subsequent log archives.

The directory and the database are not synchronized. The directory was restored from an archive file that did not reflect the ADD DBEXTENT operation; the database is also restored from that file however, the use of the changed dbextents is also restored from updates recorded in the log or log archives. Thus, the directory does not reflect the changed dbextents, but the database does.

For a DELETE DBEXTENT operation, suppose the following occurs:
1. You do a database archive
2. Later you delete dbextents
3. You attempt to do an archive restore from number 1 above.

The restore operation fails because it attempts to put data on the dbextents that have been removed.

You can prevent this problem by using the ARCHIVE or UARCHIVE option in the ADD or DELETE DBEXTENT operation. This will ensure that your current database archive reflects the changed dbextents.

The same problems occur if you use log archiving and restore the database using a database archive taken before the ADD or DELETE DBEXTENT operation. That is, if you use a back-level database archive and subsequent log archives to restore the database, the database archive that records the changes to the dbextents are skipped. For ADD DBEXTENT operations, the directory, restored from the back-level database archive, does not show the changes to the dbextents; the subsequent log archives, however, do record the use of those dbextents. Restoring the database from an old database archive and subsequent log archives can thus put the database out of synchronization with the directory. For DELETE DBEXTENT operations, the restore fails when it tries to use the removed dbextents.

If a system failure occurs during the ADD or DELETE DBEXTENT operation, restart the operation after determining and correcting the cause of the failure.

Initialization Parameters for ADD and DELETE Dbextents

Instead of using the procedure ARIS250D, you can choose to run program ARISQLDS with additional parameters. The initialization parameters that you can specify for running the ADD or DELETE DBEXTENT operation, are shown in Table 16.

Table 16. Initialization Parameters for the ADD and DELETE DBEXTENT Operation

Parameter	Default	Minimum	Maximum
DBNAME=name	SQLDS	---	---
SYSMODE=S	---	---	---
PARMID=name	None	---	---
STARTUP=E	---	---	---
DBPSWD=password	None	---	---
LOGMODE=Y|A|L|N	Y	---	---
DSPLYDEV=L|C|B	L	---	---
DUMPTYPE=P|F|N	P	---	---
TRACDBSS=nnn...	Zeros	Zeros	Twos
TRACRDS=nnnnnn	Zeros	Zeros	Twos
TRACDSC=nn	Zeros	Zeros	Twos
TRACCONV=n	Zero	Zero	Twos
TRACSTG=n	Zero	Zero	Ones
TRACEBUF=n	Zero	Zero	99999

The DBNAME, SYSMODE and STARTUP parameters are required as shown to define the run as an ADD or DELETE DBEXTENT operation. Also, PASSWORD will be required if the database VSAM data sets are password protected.

The PARMID parameter can be used to specify a source member that contains parameter specifications for the ADD or DELETE DBEXTENT operation.

The DSPLYDEV, DUMPTYPE, TRACDBSS, TRACDSC, TRACRDS, TRACCONV, TRACSTG and TRACEBUF parameters can optionally be specified. For a description of these parameters, see “Multiple User Mode Initialization Parameters” on page 47. Because ADD and DELETE DBEXTENT operations can only be done when the database manager is running in single user mode, the initialization parameters are the only means of tracing them. (Operator TRACE commands are only valid when the database manager operates in multiple user mode).

Moving Dbextents

Sometimes it may be necessary to relocate the dbextents to another device due to disk migration or to control device utilization. This is done using the VSAM BACKUP and RESTORE commands after the application server is shut down. See Figure 68 on page 154 for an example of IDCAMS BACKUP, commands and Figure 70 on page 160 for IDCAMS RESTORE commands.

Moving the Log

Sometimes you must relocate the log data set to another device because of disk migration or to control device utilization. If the following conditions are met, VSAM BACKUP and RESTORE commands can be used to make an exact copy of the original log data set and it is not necessary to reformat or reconfigure the new log data set:

- The target log data set is the identical size as the source data set
- The source log data set is not damaged.

For more information about reconfiguring or reformatting the log data set, see "Reconfiguring and Reformatting the Logs" on page 171.

Chapter 8. Making Backups and Recovering from Failures

Database recovery refers to the processing done to correct data when something goes wrong. This chapter presents a detailed description of basic recovery concepts, and how to implement them. More advanced recovery topics are discussed in Chapter 9, "Special Topics in Recovery Design," on page 167.

The problems that can occur fall into four categories:

Application Error
: Occurs when an application (for example, an ISQL command or routine, or the DBS utility) does not end successfully.

User Logic Error
: Occurs when the system or application does the requested function, but the request itself is in error — that is, the user (or application program) did not specify the correct function. For example, the user may have accidentally dropped the wrong table or dbspace.

  This is the only type of error where detection is not immediate. Therefore, it presents more of a problem. Errors in the data can go undetected for quite some time, making recovery processing very complex.

System Failure
: Occurs when the application server ends abnormally. Such failures can occur because of a severe error involving the operating system, or because of certain error conditions detected by the database manager, such as a power failure.

DASD Failure and Database Corruption
: Occurs when the database manager cannot read data from or write it to the DASD where it is stored, because the storage medium is unreadable or damaged. Such an error (also called a media failure) can occur on the log, the directory, or a data extent (DBEXTENT).

This manual discusses how to recover from system and DASD failures. Recovery from application and user logic errors is described in the *DB2 Server for VSE & VM Database Administration* manual.

There are two aspects to dealing with system and DASD failures:
- Establishing and maintaining regular recovery procedures, to ensure that you have the information available to correct the data if something goes wrong.
- Correcting the data.

Understanding Recovery Concepts

To effectively protect your data and recover it in the event of failure, you need to understand the measures built into this product. Protecting against system failures involves the *LUW*, the *log*, and the *checkpoint*. Protecting against DASD failures entails two types of archive: the *database archive* and *the log archive*.

What is a Logical Unit of Work?

The data in your database is in a *consistent* state if no changes are left only partially completed.

Some data changes cannot be expressed in only one SQL statement. For example, suppose you have a banking program to transfer money between accounts, and want to transfer $100 from a SAVINGS to a CHECKING account. The program makes this transfer in two steps:

1. Add $100 to the balance of the CHECKING account.
2. Subtract $100 from the balance of the SAVINGS account.

If the second step fails (for example, because of a system failure), the data is in an *inconsistent* state. That is, a deposit has been made to the CHECKING account, but no withdrawal has been made from the SAVINGS account.

The *logical unit of work (LUW)* prevents such inconsistencies. An LUW is a sequence of SQL statements that the system treats as a single entity. Either all the data changes made during an LUW are performed, or none is performed. In the example above, the two updates should be placed within a single LUW.

To group several SQL statements into one LUW, one uses the COMMIT WORK and ROLLBACK WORK commands.

If no problems or errors occur, the user issues the COMMIT WORK command to save all the changes made. If a problem occurs in the middle of an LUW, the user can issue the ROLLBACK WORK command to undo all the changes made since the last COMMIT WORK command.

An LUW can be as small as one SQL statement, or as large as an entire ISQL session or application execution. ISQL, by default, treats each command as an LUW, and issues a COMMIT WORK command after each SQL statement that modifies the database. Users can change this default by issuing the SET AUTOCOMMIT OFF command. For more information on the use of the AUTOCOMMIT, COMMIT, and ROLLBACK commands, refer to the *DB2 Server for VSE & VM SQL Reference* manual.

What is a Log?

The log is a file maintained on DASD that records all the changes completed by each LUW. For each change, the log records the old and new values of the updated object. If any changes to the database must be undone or redone, you can use the log to restore the data to its proper state.

In addition to the changes made by each logical unit of work, the log also records when each logical unit of work started and stopped. (It does not record logical units of work that only read information from the database).

A database must have at least one log. Optionally, you can use alternate logging to have one active log and one inactive log. If only one log is defined and the ARCHPCT value is reached, a checkpoint will occur followed by a log archive. If an inactive log is present, the database manager will attempt to switch to the inactive log once the checkpoint is complete. For more information, see “Using Alternate Logging” on page 169.

You can create an exact duplicate of your active log and inactive log by using dual logging. If dual logging is enabled and a DASD failure occurs on the active or inactive log, the database manager can continue using the backup copy. For more information, see “Using Dual Logging” on page 170.

Larger logs may be needed for tables that are being captured for DataPropagator because of the increased amount of log data written for UPDATEs to those tables which specify DATA CAPTURE CHANGES. Tables being captured will log the entire original row (not just the data that was changed), and the new data that replaces the old changed data. You should consider increasing the size of the log dbextent(s) when planning to make extensive use of this function.

What is a Checkpoint?

Checkpoints are taken periodically. During a checkpoint the database manager stops servicing users, and takes a “snapshot” of the database that includes updates from completed LUWs as well as from those that are still in progress, and writes them to DASD. In addition, a special checkpoint record is written to the log to synchronize the log with the state of the database.

What Happens after a System Failure?

Restart Recovery with a Log

If your system fails, as long as the current log is available, the database will be automatically recovered to a consistent state when you restart the application server. This process, called *restart recovery*, uses the log to ensure that changes made by LUWs are either committed (if they had successfully finished) or backed out (if they had not finished successfully).

The recovery process determines the state of each LUW; both at the time of failure and at the time of the last checkpoint before the failure. The following scenarios are shown in Figure 66 on page 146:

- LUW A: if the LUW starts and ends before the checkpoint, all the updates are safely reflected in the database at the checkpoint.
- LUW B: if the LUW starts before the checkpoint and commits work after the checkpoint but before the failure, those updates made after the checkpoint must be redone, using the log. Those updates made prior to the checkpoint are reflected in the database.
- LUW C: if the LUW starts before the checkpoint but is not completed before the failure, those updates made prior to the checkpoint must be undone using the log. The updates made after the checkpoint are not reflected in the database: thus all the updates must be re-entered.
- LUW D: if the LUW starts after the checkpoint and commits work before the failure, all its updates must be redone using the log.
- LUW E: if the LUW starts after the checkpoint and is not completed before the failure, all its updates must be re-entered since none of them are reflected in the database.

The following diagram illustrates the LUW Recovery process for the five cases described above:

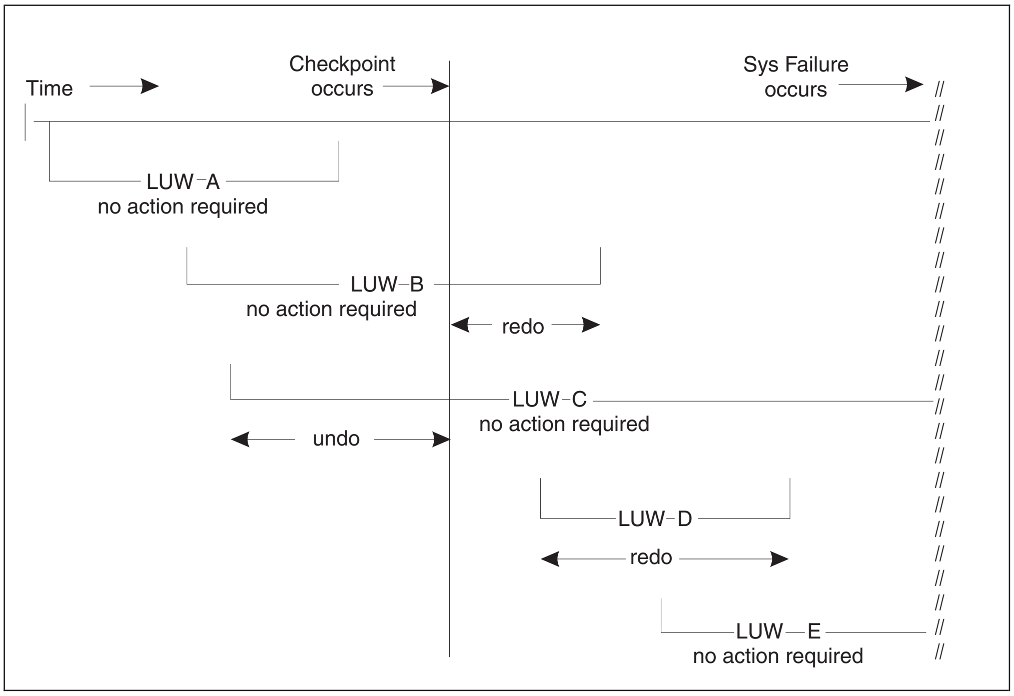


Figure 66. LUW Recovery Actions

Restart Recovery Without a Log

If the application server must be restarted without a log (due to the log either being lost, reformatted, or reconfigured immediately after the failure), the database cannot be adjusted to complete committed logical units of work or to back out uncommitted ones. In this situation, to recover the database you will have to restore a previous database archive, together with any applicable log archives.

If the database manager had been running in single user mode with LOGMODE=N, the changes made by the application are not logged. However, a checkpoint would have been taken each time the application issued a COMMIT WORK (or one was issued for the application), so most changes will have been effectively committed. Any that were uncommitted at the time of failure will be discarded when you restart the application server and will need to be re-entered.

What is an Archive?

Archiving facilities enable you to recover your database directory and storage pools from DASD failures. There are two kinds of archives: database archives and log archives.

Database Archives

A *database archive* is a tape copy of the database directory and dbextents. It can be taken using two types of facilities:

- *database manager* archiving facilities supplied with this product
- *user* archiving facilities such as VSE/VSAM Backup/Restore.

If database manager facilities are used, the database manager takes a checkpoint (the begin-archive checkpoint) and writes a copy of the database directory and the database to tape, as they were at the checkpoint. (A database archive does not include a copy of the log.) Users continue to receive service while the archive is being done.

A user archive can only be done while the application server is shut down. A user archive generally takes less time than a database manager archive.

You are not restricted to using one kind of archive for a given database; you can switch between database manager archives and user archives as often as you like. There are two situations in which the former facility is required:

- When you migrate a database between two different operating systems (for example, from VSE to VM)
- When a database archive is needed while users are accessing the database. You can avoid this situation by using log archiving (LOGMODE=L).

Experience helps you determine which method is best for you. When using any backup method, the performance improvement will be related to how full your database is. The fewer pages in your database that are allocated, the less time a database manager archive takes.

In fact, if the percent of allocated pages is low enough, a database manager archive will outperform a user archive, because the database manager only archives pages that actually contain data. User facilities archive all pages, so the time taken does not vary with the number of pages allocated.

Aside from the performance advantage that user archiving facilities may offer because they exploit particular device characteristics, consider whether your facility provides other advantages such as archiving multiple dbextents simultaneously.

For a description of how to carry out these archives, see “Performing Database Archives With Database Manager Facilities” on page 152 and “Performing Database Archives With User Facilities” on page 153.

Log Archives

A *log archive* is a copy of the log on tape. Only database manager archive facilities can be used to archive the log. Log archives can be taken either when the database manager is running or at shutdown. Because the log is usually much smaller than the database, this archive takes less time than a full database archive. For a description of how to carry it out, refer to “Performing Log Archives” on page 154.

Recovering from DASD Failures that Damage the Database

If a DASD failure occurs on one of your database devices, you can restore the database by replacing the damaged volume with a working volume (see “Replacing a Dbextent” on page 164), redefining (or restoring) the data sets on the volume, and then restoring the data from the archived database and logs (if applicable.)

There are two ways to do this. One way is to use the database archive and the active log. By loading the archive and re-applying the changes in the log, you can bring the database up-to-date because all changes made to the database since the archive are recorded in the active log. If the restore set for the database archive includes the active log, you can recover the damaged storage pools instead of the entire database using the Data Restore Feature. See the *DB2 Server for VSE & VM Data Restore* manual for more information on storage pool level recovery.

Alternatively, if you archived the log, you can use the database archive, the log archives you created since the last database archive, and your active log, to recreate the database. You would load the database archive, and reapply the changes in the log archives and the active log. If the restore set for the database archive includes the active log, you can recover the damaged storage pools instead of the entire database using the Data Restore Feature. See the *DB2 Server for VSE & VM Data Restore* manual for more information on storage pool level recovery.

The relationships among the different archives, the active log, and the current database are shown in Figure 67 on page 150. For more details, see “Restoring the Database” on page 158.

Recovering from DASD Failures that Damage a Log

If a DASD failure, such as an unresolvable I/O error, occurs on one of the log devices, there are two possibilities for recovery:

- If you are using single logging or alternate logging, replace the damaged log data set (see “Replacing a Log” on page 164), and then follow the steps in “Log Reconfiguration” on page 171. Log data from the damaged log is lost.
- If you are dual logging, replace the damaged log data set with a working data set (see “Replacing a Log” on page 164), and then start the application server with the same log mode used before the log data set was damaged. The contents of the good log data set is copied to the new log data set.

Recovering from DASD Failures that Damage the Database and Log

If a DASD failure occurs on both a database device and a log device, you can restore the database by replacing the damaged dbextent with a working data set (see “Replacing a Dbextent” on page 164), replacing the damaged log data set with a working data set (see “Replacing a Log” on page 164), and then restoring the data from the archived database and logs (if applicable) (see “Restoring the Database” on page 158).

Establishing DASD Recovery Procedures

As the system administrator, you must establish recovery procedures for your installation. The procedures you put in place will determine the degree of protection for your database. Naturally, trade-offs exist; when you allocate system resources to protect against failures, these resources are unavailable to other users. However, if a failure occurs, the recovery takes less time.

This section discusses some of the options available. Based on this information, devise a plan that best suits your requirements.

Choosing a Log Mode

One of the first decisions you must make when designing a recovery strategy is the type of *log mode* you want. The log mode is an initialization parameter that you specify when you start the application server. It has four possible values:

LOGMODE=Y
All changes to the database will be recorded in a log, but no archives of the log or database will be maintained. This value is the default. Use it if you do not need to protect your data from DASD failures. The application server will run faster, since it will not require the extra time to create archives.

LOGMODE=A
All changes to the database will be recorded in a log, and regular archives

of the database will be maintained. You can either create these archives yourself, or have them created automatically when the log reaches a certain threshold level.

LOGMODE=L
All changes to the database will be recorded in a log, and regular archives of the log will be maintained. You can either create these log archives yourself, or have them created automatically when the log reaches a certain threshold level (to prevent it from becoming too full to be effective). If alternate logging is enabled, an attempt will be made to switch to the inactive log once the active log hits the threshold. The LARCHIVE INACTIVE operator command can be used at a later time to archive the inactive log.

Log archives do not contain data, but only operations that change the database. If you use this log mode, you must take an occasional database archive as well. If a failure occurs, you can use the database archive, subsequent log archives, and the current log to recover the database.

The log archives must be continuous, recording all processing that occurred since the last database or log archive. If a gap exists, it will be impossible to restore the database to its current level. (The processing that occurred during the gap can never be reapplied to the database because it was never archived.) Gaps can occur in the sequence of log archives when, for example, you switch from LOGMODE=L to some other log mode. If the continuity of the log is broken in this manner, the database manager will force a database archive before you return to LOGMODE=L processing.

LOGMODE=N
No changes to the database are recorded. This option, which is only available in single user mode, is not recommended for normal operation but can be useful in some situations. For example, it may be more efficient not to log changes if you are loading a large amount of data into a table by using the DBS utility in single user mode. If a problem occurs while you are loading, you do not need the log to recover; you can simply start over.

Once you have decided on a log mode, use it whenever you start the application server. **Do not change it without thought and planning.** If you must do so, you may have to carry out additional procedures. For information, refer to “Switching Log Modes” on page 167.

Deciding between LOGMODE=A or L

Figure 67 on page 150 illustrates the relationships among the archives, the log, and the database when the log mode is A or L. You should consider several things before choosing one mode over the other.

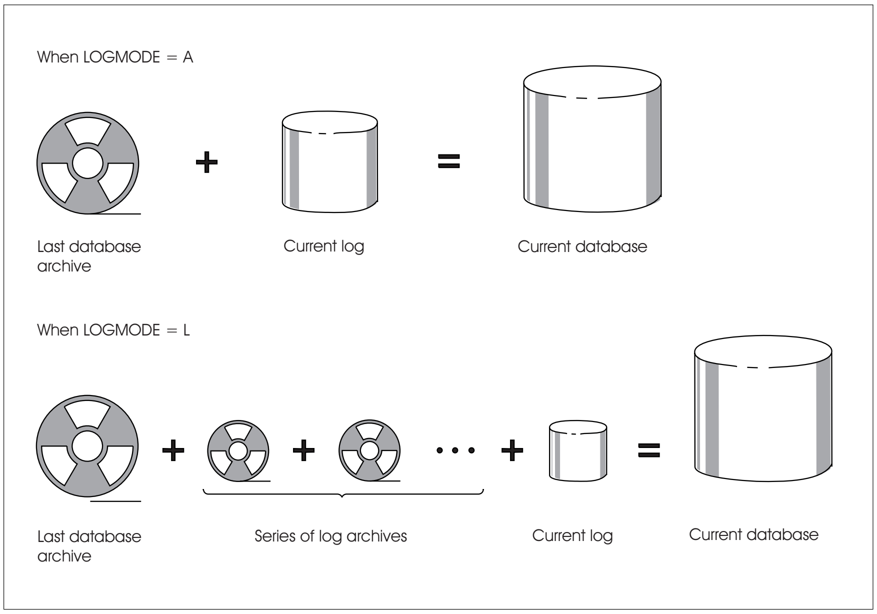


Figure 67. Relationships among the Archives, the Log, and the Database

There are three advantages to log archiving (LOGMODE=L):

- It usually takes less time, because only the log is being archived, not the directory and dbextents. This is especially helpful when the archive is being done to free log space when the database manager is running.
- Once the ARCHPCT is reached and alternate logging is enabled, an attempt will be made to switch to the inactive log. This will prevent situations where no work can continue because no one was able to load a tape for the log archive. Also, an archive of the inactive log can be done at any time via the operator command LARCHIVE INACTIVE.
- If the last database archive is unreadable or unavailable, you can bring the database back to its current status by using a back-level database manager archive or user archive, and applying to it the changes that were recorded in all subsequent logs. More recent database archives are ignored when you restore a back-level database. Two requirements must be met in order for you to use this method:
  - The log archives must be continuous. That is, you cannot have switched log modes and done a COLDLOG (with STARTUP=L) or a restore since the back-level archive was created.

    Note: You can switch from LOGMODE=L to A and then back again without breaking the continuity of the log archives, provided that no database archive was taken while LOGMODE was set to A.
  - You have not added dbspaces, added or deleted dbextents, or reconfigured the logs since the back-level database archive was made. These operations are recorded in the database directory, so if you have carried any of them out, the directory will not be synchronized with the database changes.

A disadvantage of archiving the logs is that no logical units of work can be active during the checkpoint that immediately precedes the log archive or the switch to the inactive log (if alternate logging is enabled). Concurrent access is allowed once the checkpoint is complete, but users may experience delays both before and during the checkpoint.

Another disadvantage is that it takes longer to restore the database. For example, suppose you have been taking a database archive every Friday evening and a log archive on Tuesdays and Thursdays, and on a Friday afternoon there is a media failure on the DASD that contains the database directory. You must restore the most recent database archive (from the previous Friday), and then restore the log archives from Tuesday and Thursday as well as the changes recorded in the log that was current at the time of the failure. Because only the changes to the database are stored in the log, restoring the database is similar, in processing time, to redoing all the work from the week. If there was heavy activity that week, restoration can take a long time.

Had you used database archives (LOGMODE=A) as intermediate online archives, you would only need to restore Thursday's database archive and reapply the changes on the current log. The restore time is much shorter. On the other hand, more time would have been spent doing the intermediate archives. Because media failures are infrequent, it is usually better to take intermediate log archives instead of intermediate database archives. Depending on your own experience with media failures, it may even be worthwhile to lengthen the time between database archives taken at shutdown.

Backing Up the History Area

The database manager uses the history area of the active log to keep track of recovery events (for example, database archives and log archives). The database manager can then determine which log archives belong with which database archives. If the disk containing the active log is damaged or unavailable (offsite disaster recovery scenarios), you cannot use log archives to recover the database. To avoid this situation, you should create a backup of the log file after each log archive. You can then restore this file to rebuild the log history area.

Choosing Dynamic or Static Tape Devices

Database archives or log archives can be either dynamically or statically assigned.

With dynamic tape allocation, only one tape drive can be allocated for archiving. If the archive exceeds the storage of the tape, and there is no Automatic Cartridge Loader (ACL) in use, the tape must be rewound, unloaded, and a second tape must be mounted manually. If an ACL is in use, each tape except for the last one will be unloaded and the mounting of all tapes except the first is handled by the automatic loader.

If ARCHTAPE=REW and a log archive occurs followed by a database archive, the log archive tape is left in the drive and must be unloaded manually. If ARCHTAPE=UNL, the log archive tape is unloaded upon completion of the log archive. If an ACL is present, the next tape will be mounted into the tape drive.

With static tape allocation, more than one tape drive can be allocated for archiving. The tape drives are statically assigned in the start-up JCL. If ARCHTAPE=REW, each tape of an archive is left in the tape drive and must be manually unloaded. Otherwise, if ARCHTAPE=UNL, each tape will be unloaded at the end of the tape.

The following is a sample JCL:

```
// JOB ARCHIVE
// LIBDEF *,SEARCH=(PRD2.DB2730)
// ASSGN SYS005,cuu1
// ASSGN SYS005,cuu2,ALT
// TLBL ARIARCH, ...
// TLBL ARILARC, ...
// TLBL ARILALT,...
// EXEC ARISQLDS,SIZE=AUTO, PARM='STARTUP=W,LOGMODE=L'
/*
/&
```

If there is no tape manager in use and TAPEMGR = N, you are prompted with message ARI0299A. For dynamic tape allocation, respond with the tape address cuu. If TAPEMGR=Y, the tape manager will handle the assign. For static tape allocation, respond *1*.

Archiving Procedures

This section describes how to create archives to protect your database against system failure. If a system failure occurs while you are taking an archive, see "Restarting from a System Failure While Archiving" on page 163.

Performing Database Archives With Database Manager Facilities

Database archives are tape copies of the directory and dbextents that are carried out using the database manager archiving facilities:

- By issuing an SQLEND ARCHIVE operator command, which copies the database to tape only after all LUWs complete. The copy contains all changes made by completed LUWs because no LUWs are active when the database archive is made. Log space is freed after the archive completes successfully. No changes made by incomplete LUWs are in the database archive copy. This method is preferred.

  Note: Before issuing SQLEND ARCHIVE, you should disable the DB2 Server for VSE online support by issuing the CIRT transaction. See "Online Support Considerations" on page 80.
- By issuing an ARCHIVE operator command, which lets the operator initiate a database archive at any time without either shutting down the application server or stopping access to it. The drawback, however, is that if the archive is started while applications are accessing the database, the archive copy may contain changes made by incomplete LUWs, and cannot be used for recovery from user logic errors, unless the log that was current when the database archive was taken is available. For more information about user logic errors, see the *DB2 Server for VSE & VM Database Administration* manual.

  The ARCHIVE command should be used only when you need to take a database archive to free log space but cannot afford to shut down the application server. Thus, you might want to schedule an SQLEND ARCHIVE for every Friday night, and periodic online archives during the week.

  Log space used by completed logical units of work is freed. Log space reflecting changes that are not completely included in the database archive (as of its begin-archive checkpoint) cannot be reused until the next database archive that completely includes the changes.
- By reaching the ARCHPCT value, in which case a database archive is taken automatically. The ARCHPCT initialization parameter protects the log from overflowing. (See "ARCHPCT" on page 64.) When you are running the database manager with only database archiving active (LOGMODE=A), log space that can be freed by the archive is determined by the begin-archive checkpoint and freed by the end-archive checkpoint. Log space that has been used since the longest running active logical unit of work began cannot be reused until the next

database archive is taken. If the log becomes filled to the ARCHPCT value, the database manager forces an online database archive.

Set the ARCHPCT value lower than the SLOGCUSH value, which determines when the log overflow procedure is started. When the log is filled to the percentage indicated in SLOGCUSH, the LUW that was running the longest is backed out. (Although this procedure allows the log space to be reclaimed by another forced online database archive, it can frustrate the user whose application was almost finished.)

Ideally, your log should be large enough so that the ARCHPCT value is never reached. If this value were reached at an inconvenient time (say when the operator is not at the console), database activity could stop. To prevent this from happening, you should use the ARCHIVE command to do online database archives when activity on the system is low.

Also, if you do have a database archive taken because ARCHPCT is reached, remember you cannot use this archive to recover from user logic errors. Like an online database archive initiated with the ARCHIVE command, it contains changes from incomplete LUWs, so you still need the log if this archive is the source for a restore.

Contention During an Archive

When a database archive is taken online, using database manager facilities only, other work usually continues. If, however, a condition arises during the archive that requires a checkpoint to be taken, other work must wait until the archive process completes. Such conditions include:

- A short-on-storage condition for a storage pool
- A full database log
- A COMMIT or ROLLBACK WORK statement issued during an LUW that updated data in a nonrecoverable storage pool
- An invocation of the DROP DBSPACE statement.

Note: You can use the `SHOW LOG` operator command to monitor available log space to assist you in scheduling database archives. See the *DB2 Server for VSE & VM Operation* manual for description of operator commands.

Performing Database Archives With User Facilities

User archives are database archives (LOGMODE=A or L) that are done with user facilities, such as the VSE/VSAM IDCAMS Backup/Restore feature. User archives include the database directory and all dbextents, but not the logs.

Because database manager archiving facilities are DASD-independent, they do not take advantage of particular DASD characteristics to improve performance. Some user facilities exploit these characteristics, and can archive and restore your database more quickly in some situations.

To begin archiving your database with user facilities, stop the application server and issue:

```
SQLEND UARCHIVE
```

After all logical units of work have been finished, the database manager indicates in the log history that a user archive will be taken, then prompts the operator to take the archive, and ends. (If LOGMODE=L and the log contains information, it takes a log archive before ending.) When the application server ends, the operator should take the user archive. The next time the application server is started, it displays a message to confirm that the user archive was done.

Note: Confirmation of a successful user archive is required at the next startup. If the operator specifies a restore (STARTUP=R or U) the next time the application server is started, the system assumes that the user archive was *not* taken. If the system does not prompt the operator to confirm that a user archive was created, this means that the archive was not recognized (whether or not it was successful), and it must be repeated.

Note: Do not stop the server with SQLEND QUICK and then take a user archive because the user archive will not contain consistent data.

Figure 68 shows the control statements needed to archive a database using multiple tape allocations, and using the VSE/VSAM IDCAMS command. The database in this example, has a directory called BDISK, and seven dbextents called DDSK1-DDSK7. For information about this command, see the *Using VSE/VSAM Commands and Macros* manual.

```
// JOB USERARCH
// ASSGN SYS005,181
// ASSGN SYS005,182,ALT
// DLBL IJSYSUC,'SQL301C',,VSAM
// EXEC IDCAMS,SIZE=AUTO
 BACKUP (SQL301.BDISK, -
         SQL301.DDSK1, -
         SQL301.DDSK2, -
         SQL301.DDSK3, -
         SQL301.DDSK4, -
         SQL301.DDSK5, -
         SQL301.DDSK6, -
         SQL301.DDSK7)
/*
/&
```

Figure 68. Example of VSE/VSAM BACKUP Command for a User Archive

Freeing Log Space during a User Archive

Log space is freed after a successful user archive has been confirmed at the next startup. If you take user archives and it becomes necessary to free log space when the database manager is running, you must use either the log or database archiving facilities supplied with this product to free the log space.

For log archives, set LOGMODE=L when starting the application server, and for database archives, set LOGMODE=A. In both cases, this will ensure that database archives are automatically taken if the log fills to the ARCHPCT value. Or, if you prefer to schedule your online archives yourself, periodically issue the LARCHIVE command for log archives, or the ARCHIVE command for database archives.

Note: You can use the `SHOW LOG` operator command to monitor available log space to assist you in scheduling user archives.

Performing Log Archives

A log archive is a copy on tape of all the active pages of the database log except for the last one, the log history area. To use log archiving, set LOGMODE to L. A log archive can only be performed with database manager facilities supplied with this product.

Log archives can be used with database archives taken with either database manager facilities or user facilities. Each sequence of log archives must be preceded by at least one database archive.

The log archive process can be started in the following ways:

- By issuing an SQLEND LARCHIVE operator command, which causes the database manager to copy the active log to tape when all LUWs are complete. If alternate logging is enabled, the inactive log will be archived as well, if it was not archived previously. Log space is freed after the archive completes successfully.

  Note: Before issuing SQLEND LARCHIVE, you should disable the DB2 Server for VSE online support by issuing the CIRT transaction. For more information, see "Online Support Considerations" on page 80.
- By issuing an LARCHIVE command when the database manager is running. If alternate logging is enabled, LARCHIVE will archive the inactive log as well, if it was not archived previously. This should be done when you need to take an archive to free log space but cannot afford to shut down the application server. For example, you may schedule an SQLEND ARCHIVE or SQLEND LARCHIVE for every Friday night, and schedule periodic online log archives during the week. Log space is freed after the archive completes successfully.
- By issuing an LARCHIVE INACTIVE command. This is only valid if alternate logging is enabled. This will archive the inactive log if it was not archived previously.
- By reaching the ARCHPCT value, in which case a log archive is taken automatically if single logging is used. With alternate logging, an attempt will be made to switch to the inactive log. The ARCHPCT initialization parameter protects the log from overflowing. See "ARCHPCT" on page 64. When you run the database manager with log archiving active (LOGMODE=L), log space after the begin-archive checkpoint cannot be reused until the next log archive is taken. If the log becomes filled to the ARCHPCT value, the database manager forces an online log archive. This archive cannot begin until all active logical units of work have been either committed or backed out.

  Set the ARCHPCT value lower than the SLOGCUSH value, which determines when the log overflow procedure is run and thereby protects the log from overflowing. (see "SLOGCUSH" on page 63.) When the log is filled to the percentage indicated in SLOGCUSH, the LUW that was running the longest is backed out. (Although this procedure allows the log space to be reclaimed by the online log archive, it can also frustrate the user whose application almost completed.)

  Because a log archive finishes faster than a database archive, it has less performance impact if it is done when the database manager is running. If log archives are occurring at inopportune times, however, you may want to periodically issue LARCHIVE when activity on the database manager is low. Be sure the log is large enough so the ARCHPCT limit is not reached before your scheduled log archive.
- By doing an explicit database archive while LOGMODE=L by issuing SQLEND ARCHIVE, SQLEND UARCHIVE, or ARCHIVE. Before archiving the database, the database manager does an implicit log archive (if information is in the log). If alternate logging is enabled and the inactive log was not archived yet, it will be archived at this point. Note that the database manager never does an implicit database archive.

- By restoring the database. This causes the database manager to do a log archive (if there is information in the current log) before beginning the database restore. If alternate logging is enabled and the inactive log was not archived previously, it will be archived during the restore.
- By running a COLDLOG (STARTUP=L) when alternate logging is enabled and the inactive log has not been archived. The archive is required to ensure that the inactive log information is not lost.

Contention During an Archive

When an online log archive is requested, the database manager allows any LUWs that are active to finish, but prevents any new ones from starting. A message is displayed that tells how many LUWs are active. When they are complete, the database manager takes a checkpoint and creates the log archive if single logging is used. If alternate logging is enabled, a checkpoint will occur followed by a switch to the inactive log. During the checkpoint, access to the database is disabled and any users or applications that try to start a new LUW will be in a lock wait.

You can monitor the locking contention caused by the online log archive checkpoint by using the SHOW operator commands from the operator's console. However, you cannot issue SHOW commands from ISQL to monitor the lock contention.

In most situations, only a slight delay occurs before the checkpoint is taken, but if there are long-running LUWs, it can be longer. In a worst-case scenario, a long-running LUW can delay the log archive checkpoint long enough so that the SLOGCUSH value is reached, and the database manager must roll back the longest-running LUW to free log space.

If you find that users are experiencing long delays because the database manager is trying to take a checkpoint, you can issue the SHOW operator commands to determine which user is delaying the start of the checkpoint, and then issue the FORCE command to end that user's LUW.

During the creation of the log archive of the active log, normal access to the database is usually resumed. If, however, a condition arises during the archive that requires a checkpoint to be taken, other work must wait until the archive process completes. Such conditions include:

- A short-on-storage condition for a storage pool
- A full database log
- A COMMIT or ROLLBACK WORK statement issued during an LUW that updated data in a nonrecoverable storage pool.

Note: You can use the `SHOW LOG` operator command to monitor available log space to assist you in scheduling log archives.

Labeling Your Archive Tapes

Because there are different types of archives, and each may require multiple tape volumes, it is a good idea to label the tapes externally in case you have to restore the database.

When the database manager prompts the operator to mount the tape to record the archive, it also displays a message that includes the date, time, and type of archive (database or log). For example:

```
ARI0239I External labeling of this archive is:
         Type:      log  archive
         Timestamp: 12-09-92  14:41:00
ARI0252I     Medium:    tape  183
```

The timestamp and type of archive provide identifying information about this archive, and should be written on the external label of each tape reel or cartridge. The label information is provided by the database manager for the first volume of the archive. If your archive requires more than one tape volume, add your own sequential identification to each label (for example, Tape1 of 2, Tape2 of 2).

When the database is restored, the database manager checks if there are any log archives associated with the database archive. If log archives exist, a list of them is displayed, and the time and date of each is provided. The information on the external label can be matched against this list to find the correct tapes to use for the restore.

Recovery Procedures

A system failure is any failure that causes the database manager to end abnormally. Such failures could occur because of an abnormal end of the VSE system, or because of error conditions in the database partition.

As long as the current log is available, recovery from system failures is automatic. Even if you are running the database manager in single user mode (SYSMODE=S) with no logging (LOGMODE=N), it can recover any committed updates by using the current log. Restart recovery is performed the next time the application server is started.

If there is a system failure while you are restoring the database, see "Restarting from Failure of a Database Restore" on page 161.

Restarting Procedures

To perform restart recovery procedures, the operator starts the application server with STARTUP set to one of the following values:

W	Warm start
R	Restoring from a database manager archive
F	Restoring from a database manager archive without reformatting the database data sets
U	Restoring from a user archive
S	Adding dbspaces
E	Adding or deleting dbextents
I	Reorganizing the catalog indexes
M	Catalog migration.
P	Releasing empty pages.

For all these settings, the log is checked at startup to see whether the last run of the database manager left any LUWs in progress. If it did, restart recovery processing starts and the changes made by those LUWs are backed out. Restart recovery processing also ensures that changes made by completed LUWs are, in fact, made.

Restart recovery procedures will not be performed if STARTUP is set to either **C** (for database generation) or **L** (for log reformatting or reconfiguration, called a COLDLOG operation).

For both of these settings, the database manager does not check the log, and the LUW recovery processing does not occur.

Restoring the Database

If an unresolvable I/O error occurs on any of the devices that contain the directory or dbextents, the application server ends abnormally. It may be necessary to replace the damaged volume, redefine the VSAM data sets on the new volume, and then restore the database from the most recent archive tapes.

Selecting the Archive Copy to Use

Locate the last *successful* archive of the database. If the DASD failure occurred while the most recent archive was being taken, then the last successful database archive would be the previous archive copy, not the copy interrupted by the failure.

If you are restoring from the most recent archive and the log dataset (or at least one of the log datasets in the case of dual logging) is not damaged, do not perform a COLDLOG before restoring. The active log is required for recovery. After restoring the database, follow the procedures in "Log Reconfiguration" on page 171 to recover the damaged log dataset in the case of dual logging.

If you are using a back-level database archive and LOGMODE had not been set to L when that archive was taken, or if the physical extents of the log have been changed (regardless of what LOGMODE was set to), you must run a COLDLOG with LOGMODE=Y before restoring in order to reformat the logs. Do **not** use LOGMODE=N.

You may have to redefine the directory and log datasets (or both logs in the case of dual logging) at the same time due to an I/O error. If you are restoring from a user archive, perform a COLDLOG to reformat the logs before continuing with the restore. If there is a problem with both the directory and the log, the database will have to be restored before doing the COLDLOG whether it is a DB2 Server for VSE archive or a user archive. The restore will fail when the database manager tries to read the log. After the restore fails, do a COLDLOG to reformat the logs.

If you are restoring the database by using a database archive and subsequent log archives (LOGMODE=L), locate all the necessary log archives. If the failure occurred during the archiving of the log, do not use that final log archive tape. The database manager will automatically take another log archive when it is started for the restore.

The steps to be followed to restore your database differ, depending on whether the database had been archived using database manager or user facilities.

Restoring from a Database Manager Archive

Start the application server, with STARTUP=R and LOGMODE=A or L to restore the database using an archive created with database manager facilities. The database manager prompts the operator to mount the database archive tape, and to specify on which unit (cuu) the tape is mounted. It then dynamically assigns and opens the tape, and restores the database directory and dbextents from it.

Figure 69 shows an example of doing a startup to restore a database that had been archived using database manager facilities supplied with this product.

```
// JOB RESTORE
// EXEC PROC=DBNAME01
// EXEC PROC=ARIS75PL
// TLBL ARIARCH, ...
// TLBL ARILARC, ...
// TLBL ARILALT, ...
// EXEC ARISQLDS,SIZE=AUTO,PARM='STARTUP=R,LOGMODE=L'
/*
/&
```

Figure 69. Starting with STARTUP=R to Restore a Database

Note: The ARIARCH, ARILALT, and ARILARC TLBL job control statements are not required if you have included them in your cataloged procedure for the database (DBNAME01).

It is recommended that you do not specify a VOLID parameter on TLBL statements for log archiving. Multiple log archive files can be created on a single run of the database manager. You would want these files to have different VOLIDs.

In this example, LOGMODE is set to L because the user normally uses log archiving.

Restoring from a User Archive

Shut down the application server, and restore the database using the same user facilities that created the archive.

Do not restore the database logs. If you accidentally restore the logs, the history area and all the changes to the database recorded in the log, are lost. The database manager uses the history area to track which log archives go with which database archives. For more information, see “History Area” on page 173. Even if you have been using log archiving, all changes made since the last database archive are lost. Because the history area is lost, no existing log archive can be used. To recover from accidentally restoring the log, start the application server with STARTUP=L, to do a COLDLOG to reconfigure the logs before proceeding.

After restoring the database directory and dbextents, start the application server with STARTUP=U and LOGMODE=A or L. The operator is asked whether the user restore completed successfully. If the answer is yes, then if LOGMODE=A, the changes in the log are applied to the database; if LOGMODE=L, the database manager takes an archive of the active log, and then restores the log archive tape files that are associated with the user archives. If the operator responds that the user restore was not done, the application server ends, and the operator must take the necessary action to resolve the problem.

Figure 70 shows an example of the control statements needed to perform a user restore using the VSE/VSAM Restore feature.

```
// JOB USERREST
// ASSGN SYS004,181
// DLBL IJSYSUC,'SQL301C',,VSAM
// EXEC IDCAMS,SIZE=AUTO
 RESTORE OBJECTS ((SQL301.BDISK) -
        (SQL301.DDSK1) -
        (SQL301.DDSK2) -
        (SQL301.DDSK3) -
        (SQL301.DDSK4) -
        (SQL301.DDSK5) -
        (SQL301.DDSK6) -
        (SQL301.DDSK7))
/*
/&
```

Figure 70. Example of VSE/VSAM RESTORE Command for a User Archive

In this example, VSE/VSAM restores a database having a directory (BDISK) and seven dbextents (DDSK1-DDSK7). For information about the VSE/VSAM RESTORE command, see the *Using VSE/VSAM Commands and Macros* manual.

When to Use LOGMODE=A

For both database restores (STARTUP=R) and user restores (STARTUP=U), specify LOGMODE=A when you start the application server to have the database manager restore the database without using log archive tape files. When the database is restored, the database manager applies only the changes in the active log to the database. (This is the reason you need to do a COLDLOG if you are not using the most recent database archive, or if you accidentally restored the logs during a user restore: the log does not apply to the older archive.) After completing the restore, the database manager runs with LOGMODE=A.

The database manager still checks whether there are any log archives associated with the database archive. If there are, message ARI0247D is displayed prompting the operator either to keep LOGMODE=A and restore the database without using the log archives, or to switch to LOGMODE=L and use the log archives during the restore. If the decision is made to switch to LOGMODE=L, the database manager runs as if it had been intended to do the restore with LOGMODE=L all along.

When the restore set is complete, the archive that is restored becomes the database archive for the current restore set. A restore set consists of a database archive and the log archives associated with it in the history area -- that is, those log archives that occurred between the database archive and the next restore or COLDLOG or change of log mode.

When to Use LOGMODE=L

Specify LOGMODE=L if you want the database to be restored using log archives. The database manager first restores the database archive and then takes a log archive if information is in the log that was being used when the system failed or was shut down immediately prior to the restore. It then restores the log archives that were taken after the database archive you restored. When the restore is complete, the database manager runs with LOGMODE=L.

Before restoring the database archive and each log archive, the operator is prompted to continue, stop the application server, or end the restore. Usually, the operator responds CONTINUE.

If the operator responds STOP SYSTEM, the application server ends. The next time the application server is warm-started, it will continue restoring the database using the next log archive. If it is restarted to do a restore instead of a warm start, it ignores the first restore, which was stopped, and begins a new one. If it is restarted with STARTUP=C, the application server does the equivalent of an END RESTORE (see below) and then a COLDLOG. (All subsequent log archives are no longer usable.)

The STOP SYSTEM response is used primarily for filtered log recovery. This allows you to stop the application server in the middle of a restore, change the EXTEND input file commands used for filtered log recovery, and continue the restore. For information about filtered log recovery, see the discussion on starting the application server to recover from a DBSS error in the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

The END RESTORE response is used primarily for ending a restore before processing a log archive tape that is unusable. A secondary use is to end a restore before processing a log archive that contains a user error.

Attention: If you end a restore, you may lose the ability to use subsequent log archives on a future restore.

For example, suppose you have taken a database archive and six subsequent log archives. If you discover a user error that was recorded in the fourth log archive, restore the database archive and the first three log archives. Enter END RESTORE to avoid processing the fourth, fifth, and sixth log archives. When you end the restore, it may be impossible to restore the database again using the fourth, fifth, and sixth log archives. This would be unfortunate if you had made a mistake and, in fact, should have restored the fourth log archive as well. Thus, before you respond END RESTORE, be sure you have processed the correct number of log archives.

If a situation like the one above occurs, the only way to recover the lost log archives is to restore a back-level database archive. The log archives associated with that database archive must include the ones that were lost. That is, the old database archive must have continuous log archives to the point of the END RESTORE. If it does not, you cannot recover the lost logs. For more information, see "How the History Area is Used" on page 173.

After the restore set is complete, the database archive and log archives that were just restored become the current restore set, unless the restore ended before all log archives in the restore set were applied. As a final step, the active log is restored if it directly followed the restored log archives.

Restarting from Failure of a Database Restore

Three types of errors can cause a failure of a database restore operation:

1. System failures, such as power interruptions, or operator or equipment errors that can be corrected. For example, the database manager can end because the wrong tape volume was mounted or a tape drive malfunctioned.

   In these error situations, after taking corrective action, you can restart the restore process as follows:

   - If you have received message ARI0260I (displayed at the beginning of log recovery), warm-start the application server (STARTUP=W and LOGMODE set to the value used previously). If you are using LOGMODE=L, the database manager continues with the log archive file it was processing when

  the failure occurred. A warm start saves you processing time for reading and recovering from database and log (if LOGMODE=L) archive files that have already been successfully processed.
- If you have not received message ARI0260I (or are unsure whether you have received it), restart the restore process specifying the same STARTUP and LOGMODE values you used to initiate the database restore process.

2. A log archive error that can be corrected, or a failure during UNDO/REDO processing.

   To deal with a log error that can be bypassed or corrected, refer to the section on recovering from DBSS errors in the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual, especially the discussions on UNDO and REDO processing failures during a restore.
3. A database or log archive input file error that cannot be corrected, such as a damaged archive tape volume. One of the following situations applies:
   - You were using log archiving (LOGMODE=L), and the damaged file is a database archive.

     In this situation, you can reset the database to its current state by using a previous database archive and the subsequent log archives (if there are any). You can do this only if the following conditions are met:
     - The log archives must be continuous. That is, you have not switched log modes and have not done a COLDLOG (STARTUP=L) since the previous database archive.

       Note: You can switch from LOGMODE=L to A and then back to L again without breaking the continuity of the log archives, as long as you do not take a database archive while LOGMODE is set to A. For example, suppose you accidentally start the application server with LOGMODE=A instead of L. If you immediately shut down the application server without taking a database archive, the continuity of the log archives is preserved.
     - You must not have added dbspaces, added dbextents, or reconfigured the log since the back-level database archive was made. If you have, these changes are not recorded in the log or the log archives, but are recorded in the database directory; thus, if you use the back-level database archive and subsequent log archives to restore the database, the directory will not be synchronized with the database changes, and the restore will fail.

     To reset the database using database manager facilities, restart the application server and restore the back-level database using STARTUP=R or F with LOGMODE set to L. In response to the request to mount the archive tape, mount the tape created by the previous database manager archive. When the database archive tape is restored, the operator is prompted for the subsequent log archives.

     To reset the database using user facilities, restore the database using the tape file from the previous user archive. Then start the application server with STARTUP=U and LOGMODE=L. The operator is prompted for the subsequent log archives.
   - You were using log archiving (LOGMODE=L), and the damaged file is a log archive.

     In this situation, the most current level of the database that you can restore to depends on the last undamaged log archive.

     To reset the database, restart the application server with STARTUP=W and LOGMODE set to L. The database manager tries to continue the restore by requesting the log archive that had caused the failure. (The database manager determines where it was interrupted.) Instead of responding CONTINUE, respond `END RESTORE` to the prompt in message ARI0250D.

- You were not using log archiving. The damaged tape is a database archive tape.

  Restart the application server with STARTUP=L and LOGMODE=Y (COLDLOG to reformat the logs). Then restart the restore job using a previous database archive tape.

  Note: This previous database archive must have been created by an SQLEND ARCHIVE, SQLEND UARCHIVE, or ARCHIVE command known to have been issued when no application program was accessing the database.

  In these situations, all changes made to the database since the database archive was taken are lost. You can reset the database to the consistent state that existed when that database archive tape file was created.

Restarting from a System Failure While Archiving

The procedure to recover from a system failure that occurs when the database manager is taking either a log or database archive is essentially the same as any other restart. Because it did not finish, however, the archive that was being written at the time of the failure cannot be used.

Restart the application server with STARTUP=W. If LOGMODE had been set to A or L, specify the same value; if LOGMODE had been set to Y, specify LOGMODE=A.

If the archive in-process had been an automatic archive (started by ARCHPCT), another automatic archive will be initiated immediately when the application server is started again. If it had been started by an ARCHIVE, LARCHIVE, SQLEND ARCHIVE, or SQLEND LARCHIVE command, you must reissue the command when restarting the application server. If it had been an implicit log archive created by issuing SQLEND UARCHIVE with LOGMODE set to L, reissue the SQLEND UARCHIVE command after restarting the application server with LOGMODE=L.

Restarting from Failure of a Database Generation or COLDLOG Operation

If a system failure occurs during database generation or COLDLOG processing, restart the operation after determining and correcting the cause of the failure.

In some cases, storage may need to be reclaimed before continuing processing. For example, an LUW is processing a DROP TABLE statement, a checkpoint is taken during this processing, and a COLDLOG operation immediately follows. If a media failure occurred before the COLDLOG, there is a possibility of rows from the dropped table still existing. However, the entry in the SYSTEM.SYSDROP catalog table no longer exists. To reclaim this storage, the dbspace containing this “dropped” table must be dropped before continuing processing.

Relocating the Database Manager

You can move the database manager between system DASD in two ways:

- Use VSE/VSAM Backup/Restore feature to move the database manager. For examples, see Figure 68 on page 154 and Figure 69 on page 159.
- Archive the database on the original system and restore it on the new system. For more information, see "Replacing a Dbextent."

Replacing a Dbextent

You may want to replace a dbextent because:

- You want to move your dbextents to a different device type.

  If you are replacing all the database dbextents (as you might when moving the database to a different device type), replace the log dbextents **first**. Follow the procedures in "Log Reconfiguration" on page 171.
- The dbextent is damaged because of an unrecoverable DASD error.

  You may need to replace the database directory or dbextents, because one or both were damaged. In this situation, if you are running with dual logging and only one of the logs is damaged, replace the database directory or dbextents **first** by following the steps below. Then replace the log dataset by following the procedures in "Log Reconfiguration" on page 171, and finally, restore the database by following the procedures in "Restoring the Database" on page 158.
- You want to balance your DASD workload.

  Use the instructions below if you are moving the database directory or dbextents. If you are moving your logs, refer to "Log Reconfiguration" on page 171.

To move, replace, or change a dbextent:

1. Take a database manager archive or a user archive of the database. (See "Archiving Procedures" on page 152.) The archive is required for the steps below.
2. Define the new data sets for the directory and dbextents on the new device type. Be careful when calculating their size. They should be slightly larger than the original ones, because of rounding that occurs in the space allocation algorithms. If you define the new datasets approximately equivalent in size to the old ones, the restore will probably fail.
3. Restore the database from the archive you took in Step 1. (See "Restoring the Database" on page 158.)

Replacing a Log

If you are relocating the log data sets to another device because of disk migration or to control device utilization, and the target log data set is the identical device type and size as the source log data set and the source log data set is not damaged, you can use VSE/VSAM BACKUP and RESTORE to move the log data set. See "Moving the Log" on page 141 for more information.

This section describes how to replace a log data set (DLBL LOGDSK1, LOGDSK2, ALTLGD1, and ALTLGD2). You would replace a log data set if:

1. The data set is damaged by an unrecoverable DASD error.
2. You want to change the size of your logs.
3. You want to move your data sets to a different device type.

To replace log data sets:

1. If you are replacing the active log (for single logging), either the active or inactive log (for alternate logging), or the active log and dual log (for dual

logging), take a database archive if you are running with LOGMODE=A or L, because the contents of the log, including the history area, will be lost. If you are dual logging and you are only replacing one log, the archive is not lost.

2. If you are replacing the active log (for single logging), the active or inactive log (for alternate logging), or the active log and dual log (for dual logging), follow the procedures on "Log Reconfiguration" on page 171.
3. If dual logging and you are only replacing one log, use the IDCAMS command to delete and redefine the VSAM data set for the log to be replaced. For a description of the job control statements, see Figure 86 on page 215.

Recovering to a Secondary System

To be able to recover in cases where the original database data sets are not available (for example, in an offsite disaster recovery situation), you should make a VSE/VSAM BACKUP copy of the log data set after every log archive or database archive. You would then recover to a secondary system. The secondary system must have the same dbextent configuration and number of logs as the original system.

If you have been running with LOGMODE=A and need to recover to a secondary system, do a log reconfiguration to initialize the log (see "Log Reconfiguration" on page 171), then restore the most recent archive on the secondary system.

If you have been running with LOGMODE=L and need to recover to a secondary system:

1. Do a log reconfiguration to initialize the log ("Log Reconfiguration" on page 171)
2. Use VSE/VSAM RESTORE to restore the copy of the log data set that you took after the latest database or log archive of the original system. Restore it onto the secondary system.
3. Restore the most recent archive on the secondary system.

Chapter 9. Special Topics in Recovery Design

This chapter describes how to switch log modes, how to use dual logging, how to reconfigure and reformat the logs, and how to use nonrecoverable storage pools.

Switching Log Modes

In general, you should not switch indiscriminately between log modes Y, N, L, and A: pick one mode and stick to it. However, switching to another mode may at times be required. (See "Choosing a Log Mode" on page 148 for description of log modes.)

From LOGMODE=A

To switch to LOGMODE=Y or N:

1. Issue either an SQLEND ARCHIVE or an SQLEND UARCHIVE command. With SQLEND ARCHIVE, a database archive is automatically taken, then the application server shuts down; with SQLEND UARCHIVE, the application server shuts down immediately, then you take the user archive (using your own facilities).
2. Start the application server with STARTUP=L and LOGMODE=Y to perform a COLDLOG to reformat the log.
3. Start the application server with STARTUP=W and LOGMODE=Y or N.

To switch to LOGMODE=L:

1. Issue either an SQLEND ARCHIVE or an SQLEND UARCHIVE command. With SQLEND ARCHIVE, a database archive is automatically taken, then the application server shuts down; with SQLEND UARCHIVE, the application server shuts down immediately, then you take the user archive. In either case, this database archive serves as the starting point for subsequent log archives.

   You do not have to take this database archive under either of the following two conditions:

   - You have already taken one, and have been running with LOGMODE=A since that archive.
   - You have done a restore that finished without interruption, and have done nothing to break the continuity of the restore set. (For information on how the continuity of the restore set can be broken, see "History Area" on page 173.)

   In either of these situations, the database archive you took (or restored) is in the current restore set.
2. Start the application server with STARTUP=W and LOGMODE=L.

From LOGMODE=L

To switch to LOGMODE=Y or N:

1. Shut down the application server by issuing an SQLEND LARCHIVE operator command to save the log.
2. Start the application server with STARTUP=L and LOGMODE=Y to perform a COLDLOG to reformat the log. If alternate logging is enabled and the inactive log has not been archived, you will be forced to archive the inactive log before you can reformat the logs.

3. Start the application server with STARTUP=W and LOGMODE=Y or N.

To switch to LOGMODE=A:

1. Shut down the application server by issuing an SQLEND LARCHIVE operator command to save the log.
2. Start the application server with STARTUP=W and LOGMODE=A.

   You will be warned that the continuity of the log archives will be broken.

Switching the log mode when you have been using log archiving will interrupt the continuity of the log archives, unless all you do is switch from LOGMODE=L to A and then back again without taking a database archive. (This protects you from losing a sequence of log archives if you accidentally set LOGMODE to A.) If the continuity is broken and work is done on the database, you will not be able to restore the database to its current level by using database and log archives taken prior to the break. Figure 71 shows this situation:

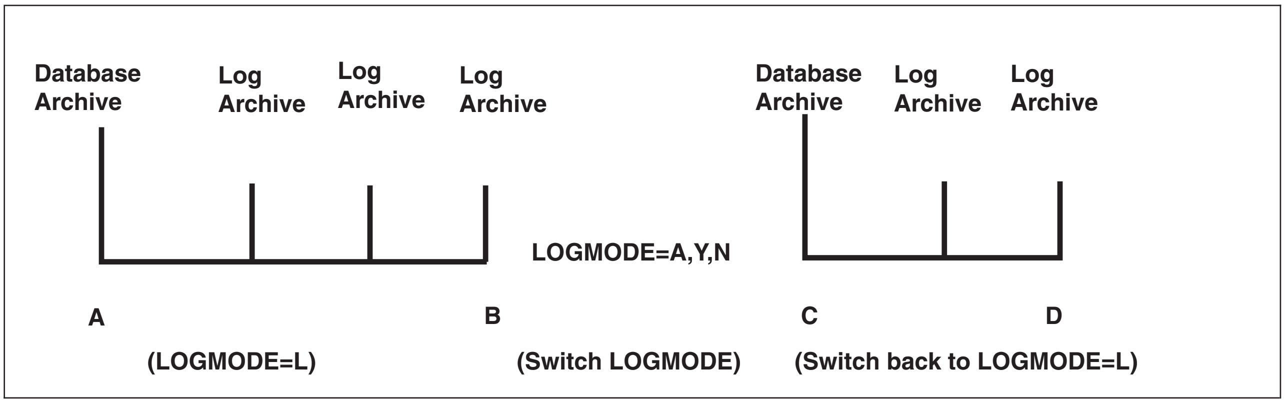


Figure 71. Log Archive Continuity

In the above diagram:

- D is the current database status.
- If you use the database archive taken at A and subsequent log archives, you can restore the database only to point B. All changes between points B and D are lost.
- If you use the database archive taken at C and subsequent log archives, you can restore the database to point D.

From LOGMODE=Y or N

To switch to LOGMODE=A:

1. Start the application server with STARTUP=W, LOGMODE=Y, and SYSMODE=M.
2. Issue either an SQLEND ARCHIVE or an SQLEND UARCHIVE command. With SQLEND ARCHIVE, a database archive is automatically taken, then the application server shuts down; with SQLEND UARCHIVE, the application server shuts down immediately, then you take the user archive (using your own facilities).
3. Start the application server with STARTUP=W and LOGMODE=A.

To switch to LOGMODE=L:

1. Start the application server with STARTUP=W, LOGMODE=Y, and SYSMODE=M.

2. Issue either an SQLEND ARCHIVE or an SQLEND UARCHIVE command. With SQLEND ARCHIVE, a database archive is automatically taken, then the application server shuts down; with SQLEND UARCHIVE, the application server shuts down immediately, then you take the user archive.

   The continuity of the log archives will have been interrupted by any work that was done while LOGMODE was set to Y or N, so you must take a new database archive. This database archive will serve as the starting point for subsequent log archives.
3. Start the application server with STARTUP=W and LOGMODE=L.

Using Alternate Logging

The alternate logging option (initialization parameter ALTLOG=Y) allows the database to switch to an inactive log when the active log is full. The switch will only occur if LOGMODE=L.

Without alternate logging, log archives can be initiated because the ARCHPCT initialization parameter is reached. This will force the operator to take an immediate log archive. However, if the operator could not respond to the prompt (for example, ARCHPCT was reached during offshift hours), no database activity can occur. Alternate logging will prevent situations like this. Here is a typical alternate logging scenario:

1. The value defined in the ARCHPCT initialization parameter is reached for the active log (call it LOGDSK1).
2. A checkpoint is taken (see "What is a Checkpoint?" on page 145). No activity can occur while the checkpoint is taking place.
3. Instead of forcing a log archive to occur immediately, we switch from the active log (LOGDSK1) to the inactive log (call it ALTLGD1). An entry is made in the history area of both the active and inactive log to document the switch. We can only switch to the inactive log if it was previously archived. If it was not archived, the database manager will immediately request an archive of both the inactive and active log disk.
4. A checkpoint is taken once the switch is complete. Once the checkpoint is complete, normal database activity can continue.
5. LOGDSK1 can be archived with the LARCHIVE INACTIVE command. The archive will not force a checkpoint. LARCHIVE INACTIVE will update the history area of both the active and inactive log to keep track of the archive of the inactive log. Also, if LOGDSK1 was not archived, the commands LARCHIVE, ARCHIVE, SQLEND LARCHIVE, SQLEND ARCHIVE, and SQLEND UARCHIVE will all archive LOGDSK1 before ALTLGD1. A restore will also force LOGDSK1 to be archived before ALTLGD1.

To establish an alternate log at database generation time, you must define two VSAM clusters of equal size for the logs. The DLBL file name for the first log must be LOGDSK1, and for the second log ALTLGD1. If dual logging is used, 2 other VSAM clusters will be needed. See "Using Dual Logging" on page 170 for further details. When starting the application server to perform database generation, specify the initialization parameter ALTLOG=Y. Later, when starting the COLDLOG operation, specify it again.

If just one log was defined at database generation time, you can use the COLDLOG operation to establish the second log to the database. Define this log to be the same size as the first, and name it ALTLGD1 on the DLBL statement.

If you have been running with LOGMODE=A or L, take a database archive or log archive before starting the COLDLOG operation. (The COLDLOG operation reformats the original log, thus erasing all the existing log data.) An example of the job control statements for starting the COLDLOG operation to create an alternate log is shown in Figure 72. Note that the LOGMODE is set to Y for a COLDLOG.

```
// JOB COLDLOG
// EXEC PROC=DBNAME01
// EXEC PROC=ARIS73PL
// EXEC ARISQLDS,SIZE=AUTO,PARM='STARTUP=L,SYSMODE=S,ALTLOG=Y,LOGMODE=Y'
/*
/&
```

Figure 72. Example of a Job Control to Start the COLDLOG Operation

Using Dual Logging

With dual logging, updates are recorded in the active log data set and its dual copy. If alternate logging is used, a dual copy of the alternate log data set is also maintained. It is unlikely that an unrecoverable DASD failure will occur on both log data sets at the same time, so this option protects you from log failures. The database manager continues running as long as it can read from and write to one of the logs. With single logging, any I/O error on the log would cause it to end.

To establish dual logging at database generation time, you must define two VSAM clusters of equal size for the logs. To establish both dual logging and alternate logging at generation time, you must define four VSAM clusters of equal size for the logs. The following table gives the required DLBL File Name for each log data set.

Table 17. Required DLBL File Name For Each Log Data Set

Log Data Set	DLBL File Name	Required When?
First log	LOGDSK1	Always Required
Dual Copy of LOGDSK1	LOGDSK2	DUALLOG=Y
Alternate log of LOGDSK1	ALTLGD1	ALTLOG=Y
Dual Copy of ALTLGD1	ALTLGD2	DUALLOG=Y & ALTLOG=Y

For the rest of this section, it will be assumed that ALTLOG=N.

When starting the application server to perform database generation, specify the initialization parameter DUALLOG=Y. Later, when starting the COLDLOG operation, specify it again.

If just one log was defined at database generation time, you can use the COLDLOG operation to establish the second log to the database. Define this log to be the same size as the first, and name it LOGDSK2 on the DLBL statement.

If you have been running with LOGMODE=A or L, take a database archive or log archive before starting the COLDLOG operation. (The COLDLOG operation reformats the original log, thus erasing all the existing log data.) An example of the job control statements for starting the COLDLOG operation to create a dual log is shown in Figure 73 on page 171. Note that the LOGMODE is set to Y for a

COLDLOG.

```
// JOB COLDLOG
// EXEC PROC=DBNAME01
// EXEC PROC=ARIS75PL
// EXEC ARISQLDS,SIZE=AUTO,PARM='STARTUP=L,SYSMODE=S,DUALLOG=Y,LOGMODE=Y'
/*
/&
```

Figure 73. Example of a Job Control to Start the COLDLOG Operation

Reconfiguring and Reformatting the Logs

During the life of a database, you may occasionally need to change the physical configuration of the logs. Such *reconfigurations* are necessary if, for example, you need to move logs from one DASD device to another.

At other times, you will need to reset the contents of the log logically. This is referred to as *log reformatting* and is required, for example, when you switch from LOGMODE=A or L to LOGMODE=Y or N.

log reconfiguration and reformatting

In this section, the term **log reconfiguration** means that the history area has been erased. **Log reformatting** means that history area has not been erased. Both erase the current database updates saved in the log.

The operation that performs log reformatting is called a COLDLOG, and is done by setting the initialization parameters STARTUP=L and LOGMODE=Y. Log reconfiguration will include the log reformatting step.

If the last shutdown was abnormal, bringing up the database using STARTUP=L and LOGMODE=Y will result in warning message ARI2010I, indicating that the current log is required for warm start or database recovery. If you have already reconfigured the log, this warning will be too late, since the log data will have been erased already.

Log Reconfiguration

Log reconfiguration erases the history area of the log. You should consider reconfiguring your log if you are using the database manager archive facility (LOGMODE=A or L) and want to do any of the following:

- Switch from single logging or alternate logging to dual logging (DUALLOG=Y)
  - this involves defining logs the same size as the first one.
- Define or remove the alternate log.
- Increase the size of your logs
  - this involves deleting the current logs and defining new, larger logs. (Note that you cannot use this process to decrease the size of your logs.)
- Change the location of your logs
  - this involves deleting the current logs and defining a set on the new devices.

To reconfigure the logs:

1. Take a database archive if you are running with LOGMODE=A or L, because the contents of the log (including the history area) will be erased.
2. Use the IDCAMS command to delete/redefine the VSAM data sets used for the logs, as indicated above. For a description of the job control statements involved, see Figure 86 on page 215.
3. Start the COLDLOG operation to reformat the log.
4. Take a new database archive, thus reflecting the new log definitions in the archive.
5. Restart the application server for normal operation.

Archiving Considerations

The continuity of log archives is broken whenever a COLDLOG reconfigure is done, so log archives taken prior to a log reconfiguration cannot be used in a restore. You should create a new archive copy of the database immediately after you complete the COLDLOG operation. This action will ensure that the archive copy of the database correctly reflects the size of the logs and whether or not dual or alternate logging is in effect.

If you use log archiving, note that a database archive, not a log archive, is needed. Reconfiguring the log breaks the continuity of the log archives, so the database archive is needed to serve as a new starting point for the log archives. (See Figure 71 on page 168.)

To create the new archive:

1. Start the application server in multiple user mode, specifying the initialization parameters STARTUP=W and SYSMODE=M. (Both of these are default values.) Set LOGMODE as you normally would (A or L).
2. After startup is complete, issue either an SQLEND ARCHIVE or an SQLEND UARCHIVE command. With SQLEND ARCHIVE, a database archive is automatically taken, then the application server shuts down; with SQLEND UARCHIVE, the application server shuts down immediately, then you take the user archive.

Log Reformatting

You must reformat the logs if you do any of the following:

- When you switch from LOGMODE=A or L to Y or N
- When you cannot do a warm start because of a logical error in the current log
- When you want to avoid log recovery in restoring a database from a back-level database archive.
- When you switch from dual logging to single logging.

To reformat the logs:

1. Take an archive if you are running with LOGMODE=A or L, because the contents of the log will be erased (but not the history area). If you are switching from LOGMODE=L to Y or N, you can take either a log archive or a database archive. If you are switching from dual logging to single logging and you use LOGMODE=L, you can take a log archive. For other log reformatting situations, take a database archive.
2. Start the application server with STARTUP=L and LOGMODE=Y to perform a COLDLOG reformat on the log.

History Area

The distinction between a log reconfiguration and log reformatting is the effect each has on an internally used portion of the log known as the *history area*. This is a portion of the log that the database manager uses to keep track of recovery events such as database archives, log archives, restores, COLDLOGs, and the switching of log modes. Log reconfiguration causes the history area to be erased; log reformatting does not.

Be aware that whenever you move the log, change its size, or delete the VSAM data set used for it, its history area is erased. If this happens, the database manager cannot tell which log archives belong with which database archives, or if the continuity of log archiving was broken. In fact, it cannot tell whether you were using log archiving at all, so it cannot allow you to restore the database using a database archive and subsequent log archives.

You can always restore the database from a back-level database archive or a current database archive, but if you lose the history area, you lose the ability to restore using any log archive that was taken before this loss. Also, if the database archive was taken online (with the ARCHIVE command), the database may be restored to an inconsistent state. For example, a LUW could have made changes before the archive was taken, and then been rolled back after the archive finished. When the database archive is restored, the changes made before the archive was taken will be in the database, but any changes made after the archive will be lost.

How the History Area is Used

The following description is not intended to be comprehensive; it only provides general background information about log archive recovery processes using the history area.

Suppose that you take a database archive (using either database manager or user facilities), followed by four log archives. The history area of the log would contain one record for each of these events:

```
Database Archive 1
Log Archive 1
Log Archive 2
Log Archive 3
Log Archive 4
```

The records in the history area itself are in an internal (unreadable) format. For ease of description, they are shown here in an externalized form.

If you now request another database archive, then because the database manager is running with LOGMODE=L, it first takes another log archive of the active log (Log Archive 5 in the example below). If alternate logging is enabled and the inactive log was not previously archived, the inactive log will be archived before the active log. For the rest of this example ALTLOG=Y. If you then take three subsequent log archives, the history area would contain the following records:

```
Database Archive 1
Log Archive 1
Log Archive 2
Log Archive 3
Log Archive 4
Log Archive 5
Database Archive 2
Log Archive 6
Log Archive 7
Log Archive 8
```

When you take an archive, the database manager generates identification information based on the processor's time-of-day clock. When you restore the database, the database manager reads this information in the database archive tape file before it looks at the history area.

During a restore, you may be prompted to take a log archive of the active log to save the changes up to the point of the restore. When you restore the database from the restore set containing this log archive (and actually restore the log archive), it is erased from the log history's restore set since it is put back into the active log.

When the database manager identifies the database archive tape that is being restored, it writes a record in the history area to indicate that a restore is being done. Next, it looks for the corresponding database archive record in the history area.

For example, suppose you start the application server with STARTUP=R, and mount the Database Archive 2 tape file. The database manager looks for the corresponding record in the history area, by searching in reverse chronological order, from the most recent to the least recent entries. When it finds it, it determines the log archives associated with the database archive by reading forward in the history area until the RESTORE record is reached. Log Archive 9 is taken before the restore set is determined. This set of records is referred to as the *restore set*.

```
                         Read back to the         Read forward to
                         Database Archive         identify associated
Write a RESTORE record:  Record:                  log records:

RESTORE                  RESTORE                  RESTORE
Database Archive 1       Database Archive 1       Database Archive 1
Log Archive 1            Log Archive 1            Log Archive 1
Log Archive 2            Log Archive 2            Log Archive 2
Log Archive 3            Log Archive 3            Log Archive 3
Log Archive 4            Log Archive 4            Log Archive 4
Log Archive 5            Log Archive 5            Log Archive 5
Database Archive 2       Database Archive 2  <--- Database Archive 2  <---
Log Archive 6            Log Archive 6            Log Archive 6       <---
Log Archive 7            Log Archive 7            Log Archive 7       <---
Log Archive 8            Log Archive 8            Log Archive 8       <---
Log Archive 9            Log Archive 9            Log Archive 9       <---
```

The database manager copies the restore set records after the RESTORE record.

```
   Database Archive 1
   Log Archive 1
   Log Archive 2
   Log Archive 3
   Log Archive 4
   Log Archive 5
   Database Archive 2  <---
   Log Archive 6       <---    Restore set
   Log Archive 7       <---
   Log Archive 8       <---
   Log Archive 9       <---
   RESTORE
   Database Archive 2  <---
   Log Archive 6       <---    Restore set copied forward
   Log Archive 7       <---
   Log Archive 8       <---
   Log Archive 9       <---
```

It then displays the restore set to the console using messages. If you restore all the log archives associated with the database archive, the history area remains as shown above, except that Log Archive 9 is erased from the restore set copied forward when it is restored to the current log. If, however, you respond END RESTORE to one of the prompts, the database manager deletes the remaining log archive records from the history area. For example, suppose you responded END RESTORE after only two of the log archives had been processed. The final two log archives in the history area are deleted:

```
Database Archive 1
Log Archive 1
Log Archive 2
Log Archive 3
Log Archive 4
Log Archive 5
Database Archive 2
Log Archive 6
Log Archive 7
Log Archive 8
Log Archive 9
RESTORE
Database Archive 2
Log Archive 6      <---   Only two log archives are restored
Log Archive 7      <---
```

When the restore is ended, processing continues and two more log archives are taken. Now the history area looks like this:

```
Database Archive 1
Log Archive 1
Log Archive 2
Log Archive 3
Log Archive 4
Log Archive 5
Database Archive 2
Log Archive 6
Log Archive 7
Log Archive 8
Log Archive 9
RESTORE
Database Archive 2
Log Archive 6
Log Archive 7
Log Archive 10     <--- New log archives
Log Archive 11     <---
```

If you must again restore the database and use Database Archive 2, the restore set will contain Log Archives 6, 7, 10, and 11. Because the database manager determines the restore set by scanning backwards in the history area until it finds a corresponding database archive record, the original Database Archive 2 record (the one before the RESTORE) is never reached. Consequently, it is impossible to use Log Archive 8 or 9 when restoring the database from Database Archive 2.

The only way to restore Log Archive 8 or 9 after you responded END RESTORE is to restore from a back-level database archive. This archive must have continuous log archives to the log archive you want to restore.

In our example, to restore the database to its status immediately before the restore, start the application server to do a restore, and restore Database Archive 1. The database manager scans backwards to the first occurrence of a Database Archive 1 record. (There is only one occurrence.) When it finds the record, it then scans forward in the history area until it either reaches the end of the history area or until it finds:

- A record that indicates that a COLDLOG was taken
- A record that indicates that LOGMODE was switched to N
- A record that indicates that LOGMODE was switched to Y
- A RESTORE record
- Two database archive records in a row (no log archive records in between)
- Records that indicate that LOGMODE was switched to A and that a database archive had been taken while LOGMODE=A. (When the database is archived, the log is reclaimed without a log archive. This breaks the continuity of the log archives.)

These records indicate a break in the continuity of the log archives. If you restore Database Archive 1 in our example, the restore set copied forward in the history area includes Log Archive 8:

```
Database Archive 1   <---
Log Archive 1        <--- New
Log Archive 2        <--- Restore
Log Archive 3        <--- Set
Log Archive 4        <---
Log Archive 5        <---
Database Archive 2   <---
Log Archive 6        <---
Log Archive 7        <---
Log Archive 8        <---
Log Archive 9        <---
RESTORE              <--- Indicates end of restore set
Database Archive 2
Log Archive 6
Log Archive 7
Log Archive 10
Log Archive 11
RESTORE
Database Archive 1   <---
Log Archive 1        <---
Log Archive 2        <---
Log Archive 3        <---
Log Archive 4        <---
Log Archive 5        <---    Restore Set Copied Forward
Log Archive 6        <---
Log Archive 7        <---
Log Archive 8        <---
Log Archive 9        <---
```

During the actual restore, only the log archives are applied. Database Archive 2 is skipped, because all the change activity is recorded in the continuous log archives.

When the database is restored, it reverts back to the state it was in before the first restore. The changes recorded in Log Archives 10 and 11 are lost.

The important points about the history area are:

1. You can issue the `SHOW LOGHIST` command to determine what log archives will be restored. To determine the restore set, scan backwards in the command output until the appropriate database archive is reached; then scan forward to determine what log archives are associated with that database archive. When you reach a recovery event that breaks the continuity of the log archives, you have reached the end of the restore set.
2. If you have responded END RESTORE and later want to restore the subsequent log archives, you must restore a back-level database archive whose associated log archives include those that were skipped by the issuing of the END RESTORE.

If the database manager cannot find the database archive in the current history area, a message is displayed saying the database archive is unknown. You are given the opportunity to do a COLDLOG (if one has not yet been done) to reformat the log. The COLDLOG is necessary because since the database manager cannot determine a recovery set, none of the log archive records in the history area applies, and hence the database manager cannot confirm that the current log applies.

The lack of a database archive record in the history area implies either that the database archive is very old, or that you have mounted the wrong database archive tape file. If you are intentionally restoring an old database archive, you must do a COLDLOG to avoid applying changes recorded in the active log.

Nonrecoverable Storage Pools

You can define storage pools that are not recoverable. Changes made to user data in nonrecoverable storage pools are not logged, which eliminates much of the overhead required for recovery operations described earlier in this chapter. Recovery is the responsibility of the user.

For some applications, the benefit derived from the reduced overhead far outweighs the effort of having to do your own recovery. The applications that benefit the most are those that do massive updating of a specific set of tables in the database. Such applications include:

- User programs that perform massive updates, using SQL INSERT, PUT, DELETE, and UPDATE statements.
- DBS utility DATALOAD and RELOAD operations involving thousands or millions of records.

If normal recovery procedures were in place, these applications would generate many log records. These not only cause processing overhead, but require a larger log, because the log must be large enough to hold all the records generated during the long-running LUW (along with the records of all other concurrent LUWs). Further, if you use archiving, the increased log activity causes more frequent archives.

For applications that cause excessive logging or archiving, you have two alternatives:

1. Run the application in single user mode with LOGMODE=N.
2. Place the tables that the application accesses in dbspaces that are assigned to nonrecoverable storage pools.

The first of these methods is usually preferable. For example, suppose you have an application that loads thousands of new records into an existing table. These records are the names and addresses of subscribers to a new monthly service that your company is offering. The data for new subscribers is loaded into the tables once a month. Between runs, users perform updates on the table using ISQL (for example, changing the address of an existing subscriber).

Now suppose you decide to run the application in single user mode with LOGMODE=N. The advantage is that after the application runs successfully and you create a database archive, the ISQL users have the benefit of full database recovery. The disadvantages are:

1. You must stop the application server to run in single user mode.

2. You must create a log or a database archive before running the application, and a database archive afterwards.
3. If LOGMODE is L, you lose the potential to restore the database to its current level by using a back-level database archive and subsequent log archives, because you have broken the continuity of the log archives.

Consider, though, the alternative of placing the data in a nonrecoverable storage pool. By doing so, you avoid having to create the archives, and you can run the application in multiple user mode and so avoid interrupting other users. However, the data is nonrecoverable. The decision depends on whether your ISQL or DBSU utility users can work without recovery. If the answer is no, or if you are not certain you can foresee all possible recovery situations, use LOGMODE=N instead.

Characteristics of Dbspaces in Nonrecoverable Storage Pools

The following discussion provides the basis for you to determine whether it is feasible to store the data for a given application in a dbspace in a nonrecoverable storage pool, and what recovery procedures you will need for such data.

There is one situation where nonrecoverable and recoverable dbspaces have the same characteristics: when the database manager is running in single user mode with LOGMODE=N. In this situation, for both types of dbspaces, if there is a failure, all updates that were committed at the time of the failure are in the database; all those that were not committed are not. This applies to any ISQL or DBS utility command that updates the database. Note that commitment includes both an explicit COMMIT command and any implicit commitment (as described earlier in this chapter).

In any mode other than LOGMODE=N, the following characteristics apply to nonrecoverable dbspaces:

- Archiving nonrecoverable dbspaces

  When you take a database archive, nonrecoverable dbspaces are archived the same way as the recoverable ones. Logging is performed differently, however, because changes to user data in nonrecoverable dbspaces are not logged.
- Locking and concurrency

  Same as for recoverable dbspaces.
- Preprocessing

  DB2 Server for VSE preprocessors never update data in nonrecoverable dbspaces.
- Atomicity of operations

  Not supported. For more information, see the discussion on the SQL statements that affect multiple rows on page 180.
- Committing work

  The database manager forces a checkpoint whenever there is an implicit or explicit COMMIT of a LUW that updated data in a nonrecoverable dbspace, to ensure that all updates in that LUW are really in the database. The checkpoint will only occur if data is modified, such as by an INSERT, UPDATE, or DELETE statement. It will not occur for LUWs that do not update data, or for data administration operations such as creating or dropping indexes or altering tables. These operations are logged and are thus recoverable.

  Thus, except when restoring from an archive (see below), a user can be sure that committed updates are in the database, and will survive a system failure or an application failure. They do not, however, survive a DASD failure unless you archive the database after the updates are made.

Note: Checkpoints cause significant system overhead and increase response time for interactive users. Thus, avoid a high frequency of LUWs that update data in nonrecoverable dbspaces. Also, a checkpoint that occurs during a database or log archive causes the database manager to end all concurrent activity until the archive is completed, so users must wait. Plan your updates to nonrecoverable dbspaces so that they do not coincide with an archive operation.

- Rolling back work

  When an LUW is rolled back (either implicitly or explicitly), the database manager does not undo successful SQL INSERT, PUT, DELETE, and UPDATE statements. Instead, it forces a checkpoint (after it rolls back any changes made to recoverable data during that LUW). This means that the nonrecoverable data appears just as though the LUW had been committed at the point when the rollback occurred.

  If you want to return the data to the state it was in before the LUW, you must undo the INSERTs, PUTs, DELETEs, and UPDATEs manually. Until you do, other users can see the uncommitted updates.

  The database manager does a checkpoint to ensure that you know what changes were made (so that you can undo them). If the checkpoint was not done, and the database manager failed before the next checkpoint, it would be difficult to tell what changes (if any) were made to the database. The checkpoint is done to make it easier for you to undo the changes.

  There are two situations where the database manager does not force a checkpoint for rollbacks of LUWs that update nonrecoverable data:

  – When it rolls back LUWs during a warm start after a system failure.

    The database manager uses the log to determine the LUWs that were in progress at the time of the failure. These LUWs are normally rolled back. Changes to nonrecoverable data are not rolled back, because they were never recorded in the log in the first place.

    There is no forced checkpoint because when the system fails, all changes made since the last checkpoint are lost. (They are not in the database.) For nonrecoverable data, in this situation, there is nothing to record at a checkpoint. For more information, see the discussion on recovering from processing failures, below.

  – When it rolls back LUWs when applying log changes during an archive restore.

    Here again, the updates are not in the log, so there is nothing to record at a checkpoint. In fact, all changes to nonrecoverable data made after the archive are lost. For more information, see the discussion on restoring from an archive, below.

  Usually the EXEC CICS ROLLBACK rolls back updates made to multiple resources, but the CICS transactions that use the two-phase syncpoint (TPSP) protocol cannot rely on this when nonrecoverable data is involved. You must make other provisions for such transactions.

- Recovering from processing failures

  Logical units of work that are in-process when a system failure occurs lose the automatic rollback that normally is done the next time the application server is started. In this situation, the state of these updates depends on when the last checkpoint occurred before the failure. Updates that were completed before the checkpoint occurred are in the database; those done after the checkpoint are not.

You must undo only the updates made by an in-process LUW that occurred before the last checkpoint. This procedure resets the data to its state before the LUW that was interrupted by the system failure.

This process applies only to nonrecoverable data. If you are also updating recoverable data in that same LUW, the normal recovery rules apply for that data.

- Restoring from an archive

  If you are restoring the database from an archive copy, all data updates to nonrecoverable dbspaces done after that archive was taken are lost. You must redo all updates since the archive to bring those dbspaces to the current level.

  Because row updates (INSERT, PUT, DELETE, UPDATE) are not recorded in the log, the filtered log recovery ROLLBACK COMMITTED WORK command does not apply. It does apply, however, for recoverable SQL statements and for the DBS utility command REORGANIZE INDEX (see below), because they are logged. For information about filtered log recovery, see the discussion on starting the application server to recover from a DBSS error in the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.

- Recoverable statements and commands

  The following SQL statements are always recoverable, even if they involve nonrecoverable dbspaces:
  - ACQUIRE DBSPACE
  - ALTER DBSPACE
  - ALTER TABLE
  - CREATE INDEX
  - CREATE TABLE
  - DROP DBSPACE
  - DROP INDEX
  - DROP TABLE

  The DBS utility command REORGANIZE INDEX is also recoverable.

  The reason these are recoverable is that the database manager does not suppress logging for them. They are logged to ensure the integrity of the database catalog tables, which always refer only to objects that exist.

  If an LUW fails to commit (implicitly or explicitly) after successfully doing any of the above statements or the command, the recovery procedures will automatically undo the statement or command. For example, suppose the following actions are in a LUW:
  1. CREATE TABLE
  2. INSERT into that table.

  If this LUW fails to be committed, the table, all its rows, and its indexes are automatically dropped from the database. Because the above statements are logged, if an LUW is committed after successfully processing the statements, they can be restored from the archive.

- Partial row updates

  Except for long strings, the problem never occurs of a single row being only partially updated (inserted, deleted, or modified). The database manager always ensures that either all processing for updating a row is in the database, or that none is. (You can get partial row updates for long strings because more than one update is needed internally for each row update you request.)

- SQL statements that affect multiple rows

  An SQL statement that causes multiple rows to be inserted, deleted, or updated can fail between row modifications, due to an error condition or a system failure. Whatever the cause, because the dbspace is nonrecoverable, some of the rows are modified in the database, and some are not.

Data That Can be Placed in Nonrecoverable Storage Pools

When you are considering placing application data in a nonrecoverable storage pool, you must determine whether the user will be able to recover it in a reasonable and relatively simple manner. If so, then the table is a candidate for a nonrecoverable storage pool.

Some examples of such data follow, along with descriptions of how to recover it, based on the rules in the previous section.

Example 1

Some applications use data that is retrieved from a source outside the database, such as VSAM data, data from another DB2 Server for VSE database, or data from a sequential file. Such tables are candidates for nonrecoverable storage pools if the following are true:

1. The data, after being loaded into database tables, is used only for read-only applications.
2. The data from the outside source is the only data in the tables. (That is, the data was not added to existing tables.)

If the application that loads the data into the database tables fails (does not COMMIT) for any reason, you can recover in either of the following ways:

- If the failing LUW included one or more CREATE TABLE statements for the tables being loaded, rerun the application. Because this statement is recorded in the log, any failure to commit would cause it to be rolled back. The table and any rows that were inserted into it would be dropped.
- If the failing LUW did not contain any CREATE TABLE statements, delete all the rows from the tables; then rerun the application step that loads the data into the tables.

If, after successfully loading the data, you restore the database from a database archive that was created before the data was loaded, the rows you loaded no longer exist in the database. You can recover as follows:

1. Bypass any steps that delete all rows from the tables or that drop and recreate the tables.

   These steps are not necessary because the database manager always records DROP and CREATE table statements in the log, even for nonrecoverable dbspaces.
2. Rerun the steps that load the data into the tables.

   You must redo the data manipulation statements (in this situation, INSERT and PUT) because they are not recorded in the log. (The restore defined the tables in the database, but did not insert any data.)

These recovery rules apply only to data that is imported and loaded once and is discarded when no longer needed. Each time the data is loaded, it completely replaces the previous version.

The key point is that the source data must exist so that it can be used to recover the read-only database version.

Example 2

Data that is retrieved from an outside source and added to existing data can also be stored in a nonrecoverable dbspace.

The data can be from any of the sources described in Example 1 above. To add it to an existing table, you could use the DBS utility DATALOAD command or an application program to perform a mass INSERT operation.

You can recover the data if each batch of added rows has a unique value in a column that identifies rows of the batch. You would need an application program that generates a unique batch identifier and places it into each record (or into each row, if the application loads the rows into a table).

If the application that loads the data fails (does not commit the work) for any reason, you can recover as follows:

1. Specify the unique values that identify the rows added to the tables.
2. Delete all the rows in tables that have these unique identifier values. These rows were inserted before the system failed.
3. Rerun the step that loads the added data into the tables.

Note: Although it is tempting to commit work frequently during loading to avoid potential recovery problems, keep in mind that the commit operations cause checkpoints, which can adversely affect overall performance.

If you restore the database using a database archive that was created before one or more of the load operations, all rows loaded since that archive no longer exist in the database.

To recover those lost rows, either:

1. Query the tables to determine the last batch of rows inserted that still exist in the database.
2. Rerun the steps that added all subsequent batches of rows to the tables.

Alternatively:

1. Delete all the rows that were loaded before the database archive was taken of the tables.
2. Reload all of the rows from the original source.

Both methods of recovery assume that the loaded data still exists somewhere outside the database, and that each batch of rows has a unique identifier.

Example 3

Read-only data that is created by one or more INSERT via subselect statements can also be stored in a nonrecoverable dbspace. For recovery to be possible, the data must be inserted into empty tables.

If the loading of the table fails to be committed, you can recover the data as follows:

1. If the LUW created the table:
   a. Recreate the table. (Because CREATE TABLE statements are always recoverable, the table is dropped when the LUW fails.)
   b. Rerun the INSERT via subselect statements to load the data.
2. If the table already exists:

   a. Delete all the rows from the table, since they reflect an incomplete update.
   b. Rerun the INSERT via subselect statements to load the data.

If you restore the database from a database archive that was created before the data was loaded, the data that was loaded is not in the database. The table is not dropped, however, even if it was created after the archive, because CREATE TABLE statements are always logged. To restore the data that was eliminated by the database restore operation:

1. If the table was created (or recreated) after the database archive, rerun the INSERT via subselect statements.
2. If the table was created before the database archive, some rows may also exist in the table. It may be impossible to identify the INSERT via subselect statements that put these rows in the table. Even if you determine the INSERT responsible for a row, it is difficult to tell if all rows originally inserted by the statement still exist. (The statement may have been in progress at the time the database archive was taken.) For this situation:
   a. Delete all rows in the table.
   b. Rerun the INSERT via subselect statements.

Avoid loading (or otherwise updating) nonrecoverable dbspaces if an online database archive could occur at the same time, because such archives typically contain changes made by incomplete LUWs. For recoverable data, this is not a problem because the log contains the rest of the changes, so when you do a restore, the archive and the log are used together to reconstruct a consistent copy of the database. For nonrecoverable data however, changes are not recorded in the log, so data can be incomplete or inconsistent because no log records are available to complete the restoration of the database.

You should also not update nonrecoverable data when an online log archive can occur, because the database manager waits until all LUWs end before creating the log archive. Because LUWs that update nonrecoverable data are usually long-running, the log archive is forced to wait. If the log fills to the SLOGCUSH point, log overflow processing will be started: this involves rolling back the longest-running LUW, which is usually the one that is updating nonrecoverable data. (For a description of the SLOGCUSH parameter, see “SLOGCUSH” on page 63.)

Data That Should Not Be Placed in Nonrecoverable Dbspaces

Any data that would be difficult or impossible for a user to recover should not be put in nonrecoverable dbspaces. Some examples are:

- Data that cannot be recreated

  This includes data whose source is destroyed after the data is loaded, and data that is manually entered into tables (with the ISQL INPUT command, for example).
- Data that is modified by application programs or terminal users after it is loaded into the database

  If the owner of the table keeps an audit trail of the updates made, you *can* put this kind of data in a nonrecoverable dbspace, and have the owner use the audit trail to do recovery. However, this is practical only if the number of updates made is small.
- Tables that are linked with referential constraints (referential integrity) to tables in recoverable dbspaces.
- Tables that are managed by DB2 Server for VSE components:

  - ISQL-stored query tables
  - ISQL-stored routine tables
  - Extract facility catalog tables
  - Other IBM-supplied tables.
- Tables with small amounts of data

  Here, recovery is not a problem. Rather, there is just not enough logging done for the data to justify the added complexity of user recovery. Let the database manager do the logging and recovery.
- Large tables where small numbers of rows are periodically added

  Here again, there is not enough logging to justify user recovery.

Setting Up Nonrecoverable Storage Pools and Dbspaces

If you want the data for a particular application to reside in a nonrecoverable storage pool, do the following:

1. Determine the dbspace requirements (size, type, and number).
2. Design a recovery scheme to use in case an LUW fails while the nonrecoverable dbspaces are being updated.
3. Design a recovery scheme to use in case restoring the database from an archive should be necessary.
4. Allocate the nonrecoverable storage pool. You can do this either during database generation, or when adding a dbextent. In either situation, use the POOL control statement (see "Adding Dbextents to a Storage Pool" on page 133).

   Attention: Once a storage pool is defined, either by adding dbextents to it or by POOL(NOLOG), you must not change it from recoverable to nonrecoverable, or the reverse.
5. Define dbspaces in this storage pool, either during database generation or when adding dbspaces (see "Adding Dbspaces to the Database" on page 125). On your control statements defining the dbspaces, specify the number of the storage pool.
6. Acquire the dbspaces you want by using the ACQUIRE DBSPACE statement. You must specify the number of the storage pool you want with the STORPOOL parameter; otherwise, the database manager will not select a dbspace from a nonrecoverable storage pool.
7. Create tables in these dbspaces. To do this, you must specify the dbspace name in the CREATE TABLE statement; otherwise, the database manager will not place a table in a nonrecoverable dbspace.

Remember to perform your recovery procedures whenever there is a LUW failure or when you must restore the database from an archive.

Querying for Nonrecoverable Storage Pools and Dbspaces

To determine whether a storage pool is nonrecoverable, issue the `SHOW DBEXTENT` operator command. The POOL NO. column shows the number of the pool. If it is positive, the storage pool is recoverable; if negative, it is nonrecoverable. For example, if the number displayed is -32, storage pool 32 is nonrecoverable; if it is 32, this storage pool is recoverable.

To determine what dbspaces are in nonrecoverable storage pools, look at the POOL column in the SYSTEM.SYSDBSPACES catalog table. If this value is positive, the pool where the dbspace is assigned is recoverable; if it is negative, the pool is nonrecoverable. Again, the absolute value of the number is the storage pool number.

Following are some sample queries you can use to determine the status of nonrecoverable storage pools and dbspaces:

- To determine which storage pools are nonrecoverable and have dbspaces assigned to them, issue:
  ```
  SELECT DISTINCT POOL -
    FROM SYSTEM.SYSDBSPACES -
    WHERE POOL > 999
  ```
  Because the data type of the POOL column is DBAHW, you specify POOL > 999 instead of POOL < `0` to retrieve the nonrecoverable (that is, negative) storage pools. The DBAHW fields do not sort the same way that SMALLINT fields do. (See the *DB2 Server for VSE & VM SQL Reference* manual for description of data types.)
- To determine how many of the public dbspaces allocated to nonrecoverable storage pool number 7 are not yet acquired, and the number of pages in each of the dbspaces, issue:
  ```
  SELECT NPAGES FROM SYSTEM.SYSDBSPACES -
    WHERE DBSPACETYPE=1 AND POOL=-7 AND OWNER='        '
  ```
  The blank OWNER column indicates that the dbspace is not yet acquired.

  To find the same information for private dbspaces, change the DBSPACETYPE value in the statement from 1 to 2.
- To determine how many storage pools remain to be defined in the database, first issue the `SHOW DBCONFIG` command to see the value of the MAXPOOLS parameter. This value, which was set during database generation, determines the maximum number of storage pools allowed.

  Next, issue `SHOW DBEXTENT` to determine the number of storage pools that are in use. Storage pools are in use only if dbextents are assigned to them. The difference between this number and MAXPOOLS is the number of pools that remain to be defined. You can define storage pools by adding extents to new pool numbers until you reach the MAXPOOLS limit.

  Alternatively, the SHOW SQLDBGEN operator command will display the current database definition, including MAXPOOLS and the assignment of dbextents to storage pools.
- To determine whether a specific table is in a nonrecoverable dbspace, issue:
  ```
  SELECT DBSPACENO FROM SYSTEM.SYSCATALOG -
    WHERE TNAME=table_name AND CREATOR=userid
  ```
  If the DBSPACENO value is `0`, the table is actually a view, and you have to query the SYSTEM.SYSVIEWS catalog table to obtain the name of the underlying table. If the DBSPACENO value is not `0`, use the value in this SELECT statement:
  ```
  SELECT POOL FROM SYSTEM.SYSDBSPACES WHERE DBSPACENO=n
  ```
  If the returned POOL value is negative, the dbspace is nonrecoverable; if it is positive, the dbspace is recoverable.

Chapter 10. Using the Accounting Facility

The accounting facility records how resources are consumed on the database manager. Resources are consumed both by individual users, and by processes that cannot be attributed to a single user, such as startup, shutdown, checkpoints, and archives. This information is collected in fixed-length records, 80 bytes long, that describe who or what consumed resources.

The records include up to 16 bytes for installation-dependent data, where you can supply information such as account numbers or project numbers. These 16 bytes can come from:

- VSE applications, provided an accounting exit has been installed as described in section "Supplying Account Numbers for Users" on page 263.
- Applications on platforms other than VM or VSE that use the DRDA protocol to connect to DB2 Server for VSE servers. In this case, 16 bytes of *user supplied data* are recorded into database manager USER accounting records. Examples of such DRDA requesters are: DB2 for OS/390 and DB2 Connect.

If you already have routines to process other accounting records, you can modify them to handle the DB2 Server for VSE records. You can also use the database manager itself to store your accounting data, and use ISQL to easily manipulate the data and generate reports.

Preparing to Use the Accounting Facility

To use the accounting facility, you must first set up the operating system, then set up the job control statements for the accounting files.

Setting Up Your System

1. IPL the VSE operating system with the JA=YES option specified on the IPL SYS command. For more information, see *VSE/ESA System Control Statements*.
2. If you have CICS, you **must** generate it with the restart resynchronization capability. (If accounting is active but restart resynchronization is not installed, the online support cannot be started; the CIRB transaction fails, and you receive an error message.) For a description of the CICS table entries required for restart resynchronization, see the *DB2 Server for VSE Program Directory*.

Setting Up a Job Control for the Accounting Files

When the VSE operating system is set up to use accounting, you need to set up a job control for the accounting files, within the job control that identifies your database. This job control must identify either one or two accounting files. It is recommended that you define two, so that you can use the *alternate accounting file support*.

If only one accounting file is used, you must shut down the database manager to process this file. With alternate accounting file support, you can switch from the current file to a second one while the database manager is running, which enables you to process the information in the first file without interrupting users. You can also use the alternate file support if there is a write error on the active file; or if you are accounting to DASD files, you can switch to the alternate file when the active file reaches the end of the extent.

If you switch to a second accounting file, you should process the closed file as soon as possible to prevent yourself from accidentally overlaying the previous session's accounting information.

The accounting files must be sequential files, and they can reside on either tape or DASD. If you define two, they must both be on the same storage medium: you cannot define one on tape and the other on DASD. If the files are on DASD, they can be native SAM files, VSE/VSAM ESDS files, or files managed by the VSE/VSAM space management for SAM feature. It is recommended that you define your accounting files on DASD as VSE/VSAM ESDS files.

Regardless of whether you use DASD or tape, you must specify the file name ARIACC1 for the first file on either the DLBL or TLBL statement. If you use two accounting files, the second file name must be ARIACC2. The database manager always opens ARIACC1 when it is started with the accounting facility active.

When the database manager ends (either normally or abnormally), it attempts to close the accounting file. If this file cannot be closed, accounting data may be lost.

Managing DASD Accounting Files

To use DASD sequential files for your accounting data, first determine the potential size of the accounting data set. Initially, you should overestimate it; then adjust it based on your experience.

To get a general idea of how many accounting records are likely to be generated, start the application server for normal multiple user mode access, and at the end of the day, issue the COUNTER BEGINLUW and COUNTER CHKPOINT operator commands. The number of accounting records generated at your installation will be smaller than, but proportional to, these values. The database manager writes an accounting record for each user on some ends of logical units of work, and on all checkpoints. Three more accounting records are written for each run: one for startup, one for operation, and one for shutdown: you can ignore these three records when making your estimate.

For example, assume your counters show that your installation does 2000 logical units of work and 200 checkpoints a day. On average, this can result in 1000 accounting records generated for users and 200 records generated for checkpoints. For environments with heavy ISQL usage, the number of records generated for users would probably be lower, while for preplanned transaction environments, it would probably be higher, so you should overestimate the number of records needed.

To get an initial estimate for the size of your accounting files, multiply your estimate of the number of records by 80 to get the approximate number of bytes. For help in determining the file size that you need, see "Storage Capacities of IBM DASD Devices" on page 341.

When you have gained experience using accounting, you can adjust your file sizes.

Files Managed by VSE/VSAM Space Management for SAM Feature: If you have the VSE/VSAM space management for SAM (sequential access method) feature, you should use it to manage your accounting files on DASD. This feature provides the following advantages:

File extendibility

It allows files to be extended by the use of the DISP=(OLD) parameter on the DLBL statement. This prevents the database manager from overlaying the accounting records generated during its previous run. Otherwise, you would have to update EXTENT job control statements over multiple runs of the database manager

Secondary allocations
: It gets up to 15 additional extents when the primary allocation is exhausted. This reduces the risk of filling the accounting file, which causes a loss of accounting data.

Monitor status
: You can monitor the status of the accounting file using the access method services LISTCAT command. You can do this without interrupting processing.

No symbolic device collision
: You do not need to worry about symbolic device address collisions, which can occur

  - when the output of both the trace and accounting facilities are directed to DASD files managed by native SAM
  - when, (in single user mode) the output of the DBS utility or the preprocessor is directed to DASD files managed by native SAM

  The collisions must be resolved in the job control statements, as described on page 191.

Figure 74 shows sample job control statements for two accounting files managed by the VSE/VSAM space management for SAM feature.

```
// DLBL ARIACC1,'ACCTFIL1',0,VSAM,DISP=(OLD,KEEP),RECORDS=(x,y),       C
              RECSIZE=80,CAT=SQLWK1C
// EXTENT ,SQLWK1
// DLBL ARIACC2,'ACCTFIL2',0,VSAM,DISP=(OLD,KEEP),RECORDS=(x,y),       C
              RECSIZE=80,CAT=SQLWK1C
// EXTENT ,SQLWK1
```

Figure 74. Job Control for DASD Accounting Files (VSAM Space Management)

Notes:

1. The DLBL file name for the primary accounting file must be ARIACC1, and for the secondary file ARIACC2.
2. The DLBL parameter VSAM indicates that these are VSAM managed files.
3. The example assumes that the files are implicitly defined to VSAM the first time they are opened.
4. Every time the application server is started, it directs output to the file identified by file name ARIACC1 (even if you are using two accounting files). To avoid having the accounting information from the previous run of the database manager erased, specify the DISP=(OLD,KEEP) option on the DLBL statement for the accounting file to indicate that the files are not to be reset at OPEN time (OLD), and are not to be deleted at CLOSE time (KEEP). This allows you to implicitly define the files the first time they are used, and to extend them (add records to them) in subsequent runs.

   If you run the database manager continually, it is advantageous to specify DISP=(NEW,KEEP) to have the accounting file erased every time it is opened. Here, you would be switching between the two accounting files, using the ALTACCT command, so that you can process the current file. If you specified DISP=(OLD,KEEP), the files would never be erased; they would keep growing.

If you do specify DISP=(NEW,KEEP), be sure to process the accounting file **immediately** after you close it. If you do not, the accounting data will be erased the next time you switch accounting files.

5. When implicitly defining the files, VSAM uses the RECSIZE and RECORDS parameters to determine how much primary and secondary space to allocate for the files.

   Set RECSIZE to 80, because that is the size of an accounting record.

   For RECORDS=(*x*,*y*), set *x* to the number of accounting records you expect to be generated during your accounting period. The value you specify for *x* is multiplied by 80 (the RECSIZE) by VSAM to determine the size of the primary space allocation. For example, if you have determined that you expect 500 accounting records to be generated for each accounting period, set *x* equal to 500. The number of bytes set by VSAM for the primary space allocation is 40000 (500 x 80). The value you specify for *y* determines the size of the secondary allocation. If you set *y* to 100, VSAM allocates 8000 bytes (100 x 80).

   If the primary allocation is full, VSE/VSAM gets up to 15 additional extents. The VSAM catalog must own sufficient unallocated space on the specified volume to satisfy the space allocation requirements for the file.

   Note: If you use an explicit definition for the VSAM clusters, then specify a maximum record size of 2000 on the RECORDSIZE parameter of the IDCAMS utility DEFINE CLUSTER command.
6. For an implicit file definition, an EXTENT statement with a volume serial number (SQLWK1 in the example) is required.
7. An ASSGN statement is not required for VSE/VSAM-managed files.

Files Managed by SAM: If you use native SAM, which cannot extend files, to manage accounting files on DASD, you must devise operating procedures to avoid overlaying the accounting information from the previous run of the database manager. This can be done by updating your job control EXTENT statements every time you start the application server, or by using two sets of job control with different EXTENT statements. When starting the application server, you would alternate between the two sets of job control statements.

If you use this approach, be certain to process the accounting file as soon as you close it; otherwise it will be overlaid the next time you start the application server by using the job control that identifies the file. If you load your accounting data into tables, consider prefixing your normal start-up job control with a DBS utility job that runs in single user mode (with ACCOUNT=N specified). This job would run the DBS utility to load accounting data from the previous session into tables to prevent it from being overlaid by the next job.

Figure 75 shows sample job control statements for native SAM accounting files.

```
// DLBL ARIACC1,'ACCTFIL1'
// EXTENT ,SQLWK1,1,0,57,38
// DLBL ARIACC2,'ACCTFIL2'
// EXTENT ,SQLWK1,1,0,95,30
// ASSGN SYS007,DISK,VOL=SQLWK1,SHR
```

Figure 75. Job Control for DASD Accounting Files (Native SAM)

Notes:

1. The DLBL file name for the primary accounting file must be ARIACC1; that for the secondary file must be ARIACC2.
2. In this example, the DASD allocation for the primary accounting file is 38 tracks on SQLWK1. The 38 tracks start at relative track 57. The secondary accounting file has an allocation of 30 tracks on the same volume starting at relative track 95.
3. If you are only using one accounting file, you must specify a new DASD allocation every time you start the application server. Otherwise, you will write over the old file. (You should also specify a different file-id.)

   Even if you are using alternate accounting files, you would have to change the extents unless you ensure that the accounting data from the previous run is processed before it is overlaid.
4. This example uses the DB2 Server for VSE default symbolic unit for DASD output (SYS007). If you do not want the default, specify the symbolic unit of your choice as the first EXTENT parameter. Also specify it in the ASSGN statement. If you are also using DB2 Server for VSE tracing with output directed to DASD, either the trace or accounting output must be directed to a symbolic unit other than SYS007. In single user mode, DBS utility output or trace output to DASD causes the same problem. You must ensure that the output from only one of the facilities is directed to SYS007.

Files Managed by VSE/VSAM ESDS: It is recommended that you define your accounting files on DASD as VSE/VSAM ESDS files because they handle the End-of-Extent situation better than VSE/VSAM managed by SAM files. All advantages of using VSE/VSAM files still remain for VSE/VSAM ESDS files, except that the files will have to be defined explicitly. Figure 76 shows sample IDCAMS commands to define two VSE/VSAM ESDS accounting files. See "Converting VSAM ESDS Accounting File Records into VSAM Managed SAM Feature Records" on page 208 for related consideration of loading the accounting records using VSE/VSAM ESDS accounting files.

```
// EXEC IDCAMS,SIZE=AUTO

DEFINE CLUSTER (NAME(ACCTFIL1) -
                 VOLUMES(SQLWK1) -
                 ORDERED -
                 REUSE -
                 RECORDS(x y) -
                 RECORDSIZE(80 80) -
                 NOINDEXED) -
                 DATA(NAME(ACCT.FILE1.DATA))

DEFINE CLUSTER (NAME(ACCTFIL2) -
                 VOLUMES(SQLWK1) -

                 ORDERED -
                 REUSE -
                 RECORDS(x y) -
                 RECORDSIZE(80 80) -

                 NOINDEXED) -
                 DATA(NAME(ACCT.FILE2.DATA))
/*
```

Figure 76. IDCAMS Commands to Define VSE/VSAM ESDS DASD Accounting Files

Note:

1. Set RECORDSIZE to (80 80) because that is the size of an accounting record.
2. For RECORDS=(*x y*), set *x* to the number of accounting records you expect to be generated during your accounting period. The value of *y* determines the size of the secondary allocation.

Figure 77 shows sample job control statements for two VSE/VSAM ESDS accounting files.

```
// DLBL ARIACC1,'ACCTFIL1',0,VSAM,                C
         CAT=SQLWK1,DISP=(OLD,KEEP)
// DLBL ARIACC2,'ACCTFIL2',0,VSAM,                C
         CAT=SQLWK1,DISP=(OLD,KEEP)
```

Figure 77. Job Control for DASD Accounting Files (VSE/VSAM ESDS)

Note:

1. The DLBL file name for the primary accounting file must be ARIACC1, and the secondary file name is ARIACC2.
2. The DLBL parameter VSAM indicates that these are VSAM managed files.
3. The example assumes that the files are explicitly defined using the sample IDCAMS commands illustrated in Figure 76 on page 191.
4. Every time the application server is started, it directs output to the file identified by file name ARIACC1 (even if you are using two accounting files). To avoid having the accounting information from the previous run of the database manager erased, specify the DISP=(OLD,KEEP) options on the DLBL statement for the accounting file to indicate that the files are not to be reset at OPEN time (OLD), and are not to be deleted as CLOSE time (KEEP).
5. If the RECSIZE parameter is specified in the DLBL statement, its value should be set to 80.

Managing Tape Accounting Files

To write accounting records to a tape file, specify a TLBL statement in your database job control. The file name on the TLBL statement must be ARIACC1. If you will be using alternate accounting files, specify a TLBL statement for a second accounting file, and call the file ARIACC2.

When accounting to tape, the database manager uses the VSE dynamic tape ASSIGN macro. The operator is prompted for the address (*cuu*) of the tape drive.

If you switch the output to the alternate file, the *cuu* of the first accounting file is unassigned by the VSE dynamic tape ASSIGN macro, and becomes available for use for any purpose, including reuse for the alternate file. Figure 78 shows an example of job control statements for two accounting files.

```
// TLBL ARIACC1,'ACCTFIL1'
// TLBL ARIACC2,'ACCTFIL2'
```

Figure 78. Example Job Control for Accounting Files on Tape

To avoid overlaying accounting information from the previous session, the database manager does not rewind the accounting tape at OPEN or CLOSE time. Therefore, when the application server is next started, another accounting file is written starting after the file from the previous session.

A block size of 2000 is used for the tape file. This provides efficient performance, and minimizes the amount of accounting data lost due to a system failure or a write error on the tape.

While it is unlikely that you will reach end-of-volume for a tape accounting file, multivolume tape support is provided. You must use only IBM standard label tape files.

Starting the Accounting Facility

To start the accounting facility, set the ACCOUNT initialization parameter to D or E to write records to disk, or T to write them to tape. If you do not want accounting, specify N (the default).

Figure 79 shows an example of a job control to start the application server in multiple use mode and to direct accounting output to DASD using VSE/VSAM ESDS accounting files. The example assumes that you have supplied appropriate job control statements for the accounting files in the database identification procedure ARIS75DB

```
// JOB SQL START
// EXEC PROC=ARIS75DB
// EXEC PROC=ARIS75PL
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='ACCOUNT=E,PARMID=WARM1'
```

Figure 79. Example Job Control to Start with Accounting Enabled

The database manager can also generate accounting records in single user mode for user programs, the DBS utility, and the preprocessors. Accounting records are not generated for:

- Log reconfigurations (STARTUP=L)
- Database generations (STARTUP=C)
- Adding dbextents (STARTUP=E)
- Adding dbspaces (STARTUP=S)
- Catalog index reorganizations (STARTUP=I)
- Releasing empty pages (STARTUP=P)
- Catalog migrations (STARTUP=M)
- PROGNAME=ARISEGB, which is the catalog update phase of an ADD DBSPACE operation.

If you specify ACCOUNT=D, E, or T in these situations, the database manager displays a warning message and ignores the ACCOUNT parameter.

To generate accounting records for single user mode programs, specify ACCOUNT=D, E, or T as you would for multiple user mode.

When the database manager ends, it closes the accounting files. You should immediately process the files to reduce the chance of overlaying them during the next run.

Operating the Accounting Facility

Little operator intervention is required to use the accounting facility. If the accounting files are on tape, the operator will be prompted to mount the tape and give the *cuu* of its drive. If they are on DASD, intervention is usually not required at all: the database manager simply opens ARIACC1, wherever it may be, and continues operation.

If you have defined two accounting files, the operator must issue the ALTACCT to close ARIACC1 and open ARIACC2 (or vise versa). If you are accounting to tape, the *cuu* for ARIACC1 becomes unassigned and available for other use, including for the alternate file; the operator is now prompted for the *cuu* of the tape drive of the alternate file. If you are accounting to DASD, the ALTACCT command does not require further operator action.

ALTACCT can be issued any number of times during a single run of the database manager. Each time, the file that is currently open is closed, and the alternate file is opened.

If the accounting file fills or a write error occurs, operator intervention is required. The operator is prompted as to whether the database manager should switch to an alternate file (if available), continue with accounting disabled, or end. If an alternate file has been defined, the operator should switch to it; if not, consider shutting down the database manager to avoid losing more accounting data. (In this situation, the accounting data in the buffer has already been lost.) The operator will need to know how to respond to this error message in advance. Other users must wait until the operator replies.

When the database manager ends (either normally or abnormally), it attempts to close the accounting file. If it cannot, some accounting data may be lost. Also, if the tape file cannot be closed, its tape mark is not written. In this situation, the operator should manually write a tape mark using the VSE WTM command. (For information on this command, see *VSE/ESA System Control Statements*.)

Whenever an accounting file is closed (either by shutting down the database manager or by switching to the alternate file), the operator should immediately process it, to reduce the risk of its being overlaid the next time the application server is started. If the file is being written to tape, overlaying can easily happen -- the database manager does not rewind the tape, but there is nothing to stop the operator from manually doing so. For DASD files, a simple job control error for VSAM-managed files can cause the error: specifying DISP=(KEEP) instead of DISP=(OLD,KEEP). For SAM-managed DASD files, it is even easier to overlay the file, as no SAM provisions exist for extending files. Thus, if the same job control is used two days in a row, the ARIACC1 file will certainly be overlaid. All of these problems can easily be avoided if the operator makes it a practice to process the file whenever it is closed.

For examples of operating procedures used in accounting, see the *DB2 Server for VSE & VM Operation* manual.

Generation of Accounting Records

Accounting records are written when one of the following occurs:

- For guest sharing, when an IUCV or APPC/VM SEVER occurs:
  - When the RELEASE option of an SQL COMMIT or ROLLBACK command is specified in multiple user mode
  - When the user ID abends
  - When the DB2 Server for VSE operator issues an SQL FORCE for the authorization id
- A user reconnects without explicitly releasing the previous session.

  For example, suppose user ID USER1 uses ISQL to implicitly connect to the database manager, does some work, and then explicitly connects as authorization ID SQLDBA to do tasks requiring DBA authority. When USER1 changes authorization IDs, the database manager writes an accounting record for authorization ID USER1 and begins a new session for authorization ID SQLDBA, even though USER1 did not explicitly release the first connection.
- The internal DB2 Server for VSE resource threshold is met or exceeded. This is checked at the end of a logical unit of work.

  For example, suppose USER1 uses ISQL with AUTOCOMMIT ON, and never issues a COMMIT or ROLLBACK WORK RELEASE. The session therefore lasts until this user reconnects or leaves ISQL. If this user works on ISQL for hours and processes many logical units of work during this long session, he or she exceeds the resource threshold a number of times. Every time this happens, an accounting record is written. Now suppose the database manager abends. The only accounting information lost is for work that USER1 did after last exceeding the threshold. If the internal threshold were not used, all accounting information about USER1's session would have been lost, which represents a significant amount of work.
- The connection between the user ID and the database manager is ended by:
  - An SQLEND QUICK command
  - A DB2 Server for VSE FORCE command
  - A CICS transaction ending.

The database manager does not write an accounting record for every logical unit of work, because too many records would be generated, resulting in high system overhead. Because most ISQL users use AUTOCOMMIT ON, practically every SQL statement issued would cause a new LUW.

Using DRDA Accounting

When a remote application requester establishes a connection with an application server, it must pass along information to uniquely identify the originating user, so that the database manager can generate proper accounting records. This passed information is the LUWID.

The format of the LUWID is shown below.

```
NETID.LUNAME.INSTANCE_NUMBER.SEQUENCE_NUMBER
```

Figure 80. DRDA LUWID

Within the VSE operating system, tracing usage to a user of local resources for billing purposes is easy because the user identification is unique. With the expansion into the SNA network, accounting poses the issue of unique site and user identification. Two pieces of information supplied by the application requester form a unique identification of a remote user:
- Access user ID of requesting application
- LUWID associated with each conversation

This information is supplied by the application requester to the application server in the SNA control structure FMH5.

Two accounting record types are used to track resource consumption of remote users using the DRDA option:
- The remote user accounting record resembles the format of an ordinary user accounting record, with certain record fields having different meaning
- The DRDA accounting record is written each time a remote user accounting record is generated

For the purpose of correlating the two records, the access user ID, DB2 Server for VSE user ID, and date/time stamp in the remote user record are duplicated in the associated DRDA record.

Supplying Accounting Data from DRDA Applications

Remote DRDA application requesters have the opportunity to send accounting information to DRDA servers using a general purpose unarchitected DRDA parameter. DB2 for MVS (Version 2 Release 3 or later) and DDCS (Version 2 Release 1) have implemented this approach for sending accounting data. Similar support was enabled for VM requesters in Version 3 Release 5.

If the database manager determines that a DRDA requester has supplied accounting data, 16 bytes of *user supplied data* is recorded into database manager USER accounting records as “installation-dependent” data. For DB2 for MVS applications, user supplied data corresponds to the MVS accounting string associated with the DB2 SQL application’s MVS address space.

For DDCS applications and DB2 CONNECT applications, user supplied data corresponds to one of the following:
- The value specified by an application with the *sqlesact()* API
- The value of the *DB2ACCOUNT* environment variable
- The value of the *DFT_ACCOUNT_STR* (default accounting string) configuration parameter.

If the DRDA protocol is used to connect VM applications to VSE servers (or any other DRDA server), user supplied data corresponds to data supplied by the ARIUXIT accounting exit described in the *DB2 Server for VM System Administration* manual.

If the database manager determines that a DRDA requester has supplied accounting data but the requester is not DB2 for MVS, DDCS or DB2 CONNECT or DB2 for VM, it inserts the string “*pppvvrrm* UNKNOWN” into USER accounting records. *pppvvrrm* is the product id (prdid) of the DRDA requester.

Note: When you are using the DRDA protocol, the installation-dependent data should conform to the following:

1. The accounting string data is converted to CCSID 500 before being sent to the DRDA server. To ensure that all characters in the string data can be represented in CCSID 500, only the characters A-Z, 0-9 and '_' (underscore) be used. If characters other than these recommended ones are used, then those characters may not translate properly when the DRDA server writes out accounting records.
2. The user-specified portion of the accounting string can be at most 16 bytes. This is true for DB2 Server for VM applications sending accounting data (which is set up in the ARIUXIT user exit) and for non-DB2 for VM DRDA requesters sending accounting data to servers.

Formats of the Accounting Records

There are four kinds of accounting records generated for users:

User records
: are generated for users on VSE who access an application server on VSE.

Remote User Records
: are generated for remote users accessing the database manager using the DRDA protocol.

DRDA records
: are generated for remote users accessing the database manager using the DRDA protocol. The database manager generates DRDA records and Remote User records for these users.

VSE guest user records
: are generated for users on VSE who access an application server on a VM operating systems. For more information, see the *DB2 Server for VM System Administration* manual.

Accounting records are also generated for system processes that cannot be attributed to a single user:

An initialization
: record is written when the application server is started. This record describes the resources consumed by the operator, checkpoint, and ready/recovery agents during the startup process.

A checkpoint
: record is written for the checkpoint agent after a checkpoint occurs. For the checkpoint that immediately follows an archive, this record reflects the resources consumed in doing the archive as well as the checkpoint.

An operation
: record is written during shutdown for the processing that the operator agent has done during the current session. (This accounting record is written only for multiple user mode, as operator communications are not possible in single user mode.)

A termination
: record is written that summarizes the resources consumed during the current session.

Note: Internal resource thresholds are not used for system processes.

Initialization Records

```
Columns:
1       9       17      25                41          53  57  61  65  69 73 75  79
|       |       |       |                 |           |   |   |   |   |  |  |   |

SQLDBA  SQL/DS  INIT                      051389182005                   19 ISQL
```

Column	Data Type	Description
1-8	CHAR (8)	Jobname of the database partition
9-16	CHAR (8)	“SQL/DS ”
17-24	CHAR (8)	“INIT ”
25-40	CHAR (16)	Reserved (blanks)
41-52	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS). This format may also be DDMMYYHHMMSS. The format is controlled by the DATE parameter of the VSE STDOPT job control command or statement
53-56	CHAR (4)	Blank
57-60	CHAR (4)	Blank
61-64	INTEGER	Duration of the startup process (in seconds)
65-68	INTEGER	Processor time used by the startup process (in 300ths of a second)
69-72	INTEGER	Number of times the database manager looked at a page buffer during startup (equivalent to issuing COUNTER LPAGBUFF immediately after startup)
73-74	CHAR (2)	Century number of Date (’19’ or ’20’)
75-78	CHAR (4)	The xSQL record identifier, where x = I for Initialization
79-80	CHAR (2)	Reserved (blanks)

Operator and Checkpoint Records

```
Columns:
1       9       17      25                41          53  57  61  65  69  73 75  79
|       |       |       |                 |           |   |   |   |   |   |  |   |

SQLDBA  SQL/DS  SYSTEM                    051389182005            0083032819 CSQL
```

Column	Data Type	Description
1-8	CHAR (8)	Jobname of the database partition
9-16	CHAR (8)	“SQL/DS ”
17-24	CHAR (8)	“SYSTEM ”
25-40	CHAR (16)	Reserved (blanks)
41-52	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS). This format may also be DDMMYYHHMMSS. The format is controlled by the DATE parameter of the VSE STDOPT job control command or statement.
53-56	CHAR (4)	Blank
57-60	CHAR (4)	Blank
61-64	INTEGER	Binary zero
65-68	INTEGER	Processor time used (in 300ths of a second)

Column	Data Type	Description
69-72	INTEGER	Number of times this agent looked at a page buffer (equivalent to issuing COUNTER LPAGBUFF for only this agent)
73-74	CHAR (2)	Century number of Date ('19' or '20')
75-78	CHAR (4)	The xSQL record identifier, where x = C for Checkpoint or O for Operator).
79-80	CHAR (2)	Reserved (blank)

Termination Records

```
Columns:
1       9       17      25               41          53  57  61  65  69 73 75  79
|       |       |       |                |           |   |   |   |   |  |  |   |

SQLDBA  SQL/DS  TERM                     051389182005                   19 TSQL
```

Column	Data Type	Description
1-8	CHAR (8)	Jobname of the database partition
9-16	CHAR (8)	"SQL/DS "
17-24	CHAR (8)	"TERM "
25-40	CHAR (16)	Reserved (blanks)
41-52	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS). This format may also be DDMMYYHHMMSS. The format is controlled by the DATE parameter of the VSE STDOPT job control command or statement.
53-56	CHAR (4)	Blank
57-60	CHAR (4)	Blank
61-64	INTEGER	Time, in seconds, from startup to shutdown
Note: The following are totals for the entire run of the database manager that are extracted from the data that is used by the COUNTER command.		
65-68	INTEGER	DASDIO - Total number of DASD I/Os
69-72	INTEGER	LPAGBUFF - Number of times the database manager looked at a page buffer
73-74	CHAR (2)	Century number of Date ('19' or '20')
75-78	CHAR (4)	The xSQL record identifier, where x = T for Termination
79-80	CHAR (2)	Reserved (character blanks)

User Records

```
Columns:

1       9       17      25               41          53  57  61  65  69 73 75  79
|       |       |       |                |           |   |   |   |   |  |  |   |

SQLDBA  JOB1    MYID    USER DATA HERE   051389182005BADDEBTS           19 USQL
```

Column	Data Type	Description
1-8	CHAR (8)	Jobname of the database partition

Column	Data Type	Description
9-16	CHAR (8)	For batch and VSE/ICCF environments: the jobname of the user partition. For online environments: blanks. (The example record above is for batch environments.)
17-24	CHAR (8)	DB2 Server for VSE authorization ID that was established, implicitly or explicitly, using the connect process
25-40	CHAR (16)	If you wrote your own ARIUXIT exit to generate installation-supplied data, this data is placed here for batch/ICCF and CICS applications. If you did not write such an exit, this contains character blanks for batch/ICCF applications. For CICS applications, the following information is put in the field: **25-28** CICS transaction ID **29-36** CICS signon ID (if available) **37-40** CICS terminal ID (if available)
41-52	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS). The format can also be DDMMYYHHMMSS. The format is controlled by the DATE parameter of the VSE STDOPT job control command or statement.
53-60	CHAR (8)	The name of the package that was last active for the application
Note: The following are totals for the agent. They show values accumulated for a user.		
61-64	INTEGER	Active time (that is, time that the user was connected to an agent) in seconds
65-68	INTEGER	Processor time used (in 300ths of a second)
69-72	INTEGER	Number of times this agent looked at a page buffer (this value is equivalent to the LPAGBUFF counter value for an individual user)
73-74	CHAR (2)	Century number of Date ('19' or '20')
75-78	CHAR (4)	The xSQL record identifier, where x = U for User
79-80	CHAR (2)	Reserved (character blanks)

Remote User Records

```
Columns:
1       9       17      25                41          53  57  61  65  69 73 75  79
|       |       |       |                 |           |   |   |   |   |  |  |   |
SQLDBA  JOB1    MYID    USER DATA HERE    051389182005BADDEBTS            19USQL
```

Column	Data Type	Description
1-8	CHAR (8)	Jobname of the database partition (application server)
9-16	CHAR (8)	Access user ID of the application or interactive user (application requester)
17-24	CHAR (8)	DB2 Server for VSE authorization ID that was established, implicitly or explicitly, using the connect process
25-40	CHAR (16)	If your installation has an accounting exit that uses these bytes, this area is filled with installation-supplied data. For more information, see "Supplying Accounting Data from DRDA Applications" on page 196.
41-52	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS)
53-60	CHAR (8)	The name of the package that was last active for the application
Note: The following are totals for the agent. They show values accumulated for a user.		
61-64	INTEGER	Active time (that is, time that the user was connected to an agent) in seconds
65-68	INTEGER	Processor time used (in 300ths of a second)

Column	Data Type	Description
69-72	INTEGER	Number of times this agent looked at a page buffer (this value is equivalent to the LPAGBUFF counter value for an individual user)
73-74	CHAR (2)	Century number of Date ('19' or '20')
75-78	CHAR (4)	The xSQL record identifier, where x = U for User
79-80	CHAR (2)	Reserved (character blanks)

DRDA Records

```
Columns:
1       9       17      25          37                          64     73 75  79
|       |       |       |           |                           |       | |   |
SQLDBA  JOB1    MYID    051389182005nnTORONET.SP6AGATnnnnnnnn          19RSQLC0
```

Column	Data Type	Description
1-8	CHAR (8)	Jobname of the database partition (application server)
9-16	CHAR (8)	Access user ID of the application or interactive user (application requester) accessing the application server
17-24	CHAR (8)	DB2 Server for VSE authorization ID that was established, implicitly or explicitly, using the connect process
25-36	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS)
37-63	CHAR(27)	LU 6.2 LUWID. This field is composed of the following subfields: **37-37** Length of the entire LUWID: a 1-byte binary integer **38-38** Length of the qualified LUNAME: a 1-byte binary integer **39-*n*** Qualified LUNAME (NETID.LUNAME): a character subfield in which *n* depends on the length value in column 38 **(*n*+1)-(*n*+6)** Instance number: a bit data field **(*n*+7)-(*n*+8)** Sequence number: a bit data field If the LUWID is less than 25 bytes, the remaining columns are padded with blanks
64-72		Reserved
73-74	CHAR (2)	Century number of Date ('19' or '20')
75-78	CHAR (4)	The xSQL identifier to separate the DB2 Server for VSE accounting records from other VSE accounting records, where x = R for remote user
79-80	CHAR (2)	Reserved (character blanks)

Notes:

1. DB2 Server for VSE does not provide any data on costs incurred in communications.
2. The remote user accounting record column 9-16 has the remote application requesters access user ID. For a local batch job, this field contains the jobname.
3. The remote user accounting record column 25-40 may contain system dependant information for the application requester. For a local batch job, this field contains data retrieved from the accounting exit.

VSE Guest User Records

```
Columns:
1       9       17      25              41          53  57  61  65  69 73 75  79
|       |       |       |               |           |       |   |   |   | |   |
SQLDBA  VSEMCH1 MYID    USER DATA HERE  051389182005DEBTS               19USQLC0
```

Column	Data Type	Description
1-8	CHAR (8)	VM user ID of the database machine (fixed by CP)
9-16	CHAR (8)	For batch and VSE/ICCF environments, the jobname of the user partition. For online environments, the VM user ID of the VSE machine. (The example record above is for online environments.)
17-24	CHAR (8)	DB2 Server for VSE connected authorization ID that was established using the connect process (this can be an explicit or implicit connection)
25-40	CHAR (16)	Installation-supplied data. If you are in a batch or VSE/ICCF environment, and have not coded an accounting exit that supplies information to this field, the database manager leaves character blanks. In an online environment, if you have not coded an accounting exit to supply the information, the following is put in the field: **25-28** CICS transaction ID **29-31** CICS terminal operator ID (if available) **32-35** CICS terminal ID (if available) **36-39** This field contains character blanks, unless you have coded your own cancel exit. For information on cancel exits in VSE, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual. **40** Blank
41-52	CHAR (12)	Date and time of the accounting record (MMDDYYHHMMSS)
53-60	CHAR (8)	The name of the package that was last active for the application (also referred to as prepname or program name)
Note: The following are totals for the agent. They show values accumulated for a user.		
61-64	INTEGER	Active time (the time that the user was connected to an agent) in seconds
65-68	INTEGER	Processor time used (in milliseconds). In the VSE guest user accounting record passed to VM/ESA, processor time is recorded in thousandths of a second (milliseconds).
69-72	INTEGER	Number of times this agent looked at a page buffer (equivalent to the LPAGBUFF counter value for an individual user)
73-74	CHAR (2)	Century number of Date ('19' or '20')
75-78	CHAR (4)	The xSQL identifier to separate DB2 Server for VSE accounting records from other VM accounting records, where x = U for User.
79-80	CHAR (2)	Record identifier (character X'C0') fixed by CP

Maintaining Accounting Data

Accounting data, like any other data, can be loaded into tables and maintained by any DB2 Server for VSE facility. The following sections describe how to set up dbspaces to hold accounting records and present an example. You will have to modify the example tables to meet your own installation's requirements.

Setting up a database for accounting data involves the same activities that would be done for any data application:

1. Adding and acquiring a dbspace
2. Creating tables for the accounting data
3. Creating views on those tables
4. Creating indexes on those tables.

Considerations for an Accounting Dbspace

Because accounting data is usually read-only, it is most suited for a private dbspace. When it is in a private dbspace, multiple users are able to read it as long as the tables are not being loaded. (If they are being loaded, users get an immediate notification that a load is taking place in the form of a negative SQLCODE).

Also, because the data is read-only and because its source is a sequential file, it is a candidate for a nonrecoverable dbspace. For information on the advantages and disadvantages of this type of storage, see "Nonrecoverable Storage Pools" on page 177.

The size of the dbspace depends on a number of factors. The key considerations are:
- The number of accounting records you want to keep online
- The row length of the records
- The index space requirements.

When you have determined these factors, you can estimate the size of the dbspace needed by using the formulas in Appendix B, "Estimating Database Storage," on page 341.

To estimate the rate at which your installation generates accounting records, use the accounting facility for a trial period (a day or a week). Or, you can try to make an initial estimate using the method shown on page 188.

Tables to Hold Accounting Data

One approach to organizing accounting records is to place them in four separate tables:
- One to hold the termination records, which summarize the resources consumed during an entire session of the database manager.
- One to hold the initialization, operator, and checkpoint records, which describe the overhead resources consumed by the database manager processes.
- One to hold user records, which describe the resources consumed by individual users.
- One to hold remote access records, which contain the LUWID. The records also contain the user ID and datetime value that can be used to match with the regular user records.

Figure 81 shows the statements you could issue to create these three tables, here named SQLDETAIL (for termination records), SYSDETAIL (for initialization, operator and checkpoint records), and USERDETAIL (for user records).

```
CREATE TABLE SQLDETAIL(SQLNAME  CHAR(8),
                       DATE     CHAR(6),
                       TIME     CHAR(6),
                       RUNTIME  INTEGER,
                       DASDIO   INTEGER,
                       LPAGBUFF INTEGER,
                       CENTURY  CHAR(2) ) IN SQLDBA.ACCTNG;
```

Figure 81. Example of DBS Utility Commands to Create Accounting Tables (Part 1 of 4)

```
CREATE TABLE SYSDETAIL(SQLNAME  CHAR(8),
                       TYPE     CHAR(8),
                       DATE     CHAR(6),
                       TIME     CHAR(6),
                       RUNTIME  INTEGER,
                       CPUTIME  INTEGER,
                       LPAGBUFF INTEGER,
                       CENTURY  CHAR(2) ) IN SQLDBA.ACCTNG;
```

Figure 81. Example of DBS Utility Commands to Create Accounting Tables (Part 2 of 4)

```
CREATE TABLE USERDETAIL(SQLNAME CHAR(8),
                       USERPART   CHAR(8),
                       SQLUSER  CHAR(8),
                       USERDATA CHAR(16),
                       DATE     CHAR(6),
                       TIME     CHAR(6),
                       PNAME    CHAR(8),
                       ATIME    INTEGER,
                       CPUTIME  INTEGER,
                       ULPAGBUF INTEGER,
                       CENTURY  CHAR(2) ) IN SQLDBA.ACCTNG;
```

Figure 81. Example of DBS Utility Commands to Create Accounting Tables (Part 3 of 4)

```
CREATE TABLE DRDADETAIL(SQLNAME  CHAR(8),
                        ACCUSRID CHAR(8),
                        SQLUSER  CHAR(8),
                        DATE     CHAR(6),
                        TIME     CHAR(6),
                        LUWID    VARCHAR(27),
                        CENTURY  CHAR(2)) IN SQLDBA.ACCTNG;
```

Figure 81. Example of DBS Utility Commands to Create Accounting Tables (Part 4 of 4)

Note: If you have accounting tables defined from an earlier release, you can use the ALTER TABLE statement to add the CENTURY column to your existing tables.

The information for all the columns in the tables is loaded directly from the accounting records. These tables are described in detail below.

SQLDETAIL Table

Each row of the SQLDETAIL table contains selected data from one termination accounting record, and represents one session of the database manager. The following information is inserted into the SQLDETAIL columns:

SQLNAME	The jobname of the database partition
DATE	The dates from the termination records
TIME	The times from the termination records
RUNTIME	The time, in seconds, from startup to shutdown
DASDIO	The total number of DASD I/Os for the database manager session
LPAGBUFF	The total number of times that the database manager looked at a page buffer
CENTURY	The century numbers of the dates from the termination records.

SYSDETAIL Table

Each row of the SYSDETAIL table contains selected data from one initialization, operator or checkpoint accounting record. The following information is inserted into its columns:

SQLNAME	The jobname of the database partition
TYPE	INIT is inserted if the row describes an initialization record, and SYSTEM is inserted if the row describes an operator or checkpoint record
DATE	The dates from the operator/checkpoint or initialization records
TIME	The times from the operator/checkpoint or initialization records
RUNTIME	If the value in TYPE is INIT, this value shows the amount of time for the initialization process to finish (in seconds); if the value in TYPE is SYSTEM, this value contains binary zeros
CPUTIME	The processor time used (in 300ths of a second)
LPAGBUFF	The number of times the agent (represented by the accounting record) looked into a page buffer
CENTURY	The century numbers of the dates from the initialization records.

USERDETAIL Table

Each row of the USERDETAIL table contains selected data from one user accounting record, either from a local or a remote processor. The following information is inserted into its columns:

SQLNAME	The jobname of the database partition
USERPART	The jobname of the user partition for batch/ICCF (for rows that describe the accounting information for online users, USERPART is blank)
SQLUSER	The authorization ID that was established, explicitly or implicitly, during the connect process
USERDATA	The installation-supplied accounting data. If you have not coded an accounting exit, this column contains blanks for rows that contain accounting data for batch/ICCF users; for online users, other information is displayed. For more information, see the description of the user accounting records on page 199.

DATE	The dates from the user records
TIME	The times from the user records
PNAME	The name of the package that was last active for the application
ATIME	The active time (that is, the time that the user was connected to an agent) in seconds
CPUTIME	The processor time used (in 300ths of a second)
ULPAGBUF	The number of times the agent looked into a page buffer.
CENTURY	The century numbers of the dates from the user records.

DRDADETAIL Table

Each row of the DRDADETAIL table contains selected data from DRDA accounting records. The columns are described as follows:

SQLNAME	The jobname of the data partition (application server)
ACCUSRID	The access user ID of the application or interactive user (application requester) accessing the application server
SQLUSER	The authorization ID that was established, explicitly or implicitly, during the connect process
DATE	The dates from the DRDA accounting records
TIME	The times from the DRDA accounting records
LUWID	The qualified LUNAME, the sequence number, and the instance number
CENTURY	The century numbers of the dates from the DRDA accounting records.

Loading the Accounting Data

If you have created the tables described above, you can use the DBS utility to load the accounting records into the tables. For example, the commands shown below load the tables and list their contents. (The example shows ARIACC1 as the input file. Use ARIACC2 if you are loading records from the alternate accounting file.)

```
CONNECT SQLDBA IDENTIFIED BY SQLDBAPW;
SET ERRORMODE CONTINUE;
 DATALOAD TABLE(SQLDETAIL) IF POS(75-78)='TSQL'
   SQLNAME   1-8  CHAR
   DATE     41-46 CHAR
   TIME     47-52 CHAR
   RUNTIME  61-64 FIXED
   DASDIO   65-68 FIXED
   LPAGBUFF 69-72 FIXED
   CENTURY  73-74 CHAR  NULL IF POS(73-74) = 0
 DATALOAD TABLE(SYSDETAIL) IF POS(75-78)='ISQL'
   SQLNAME   1-8  CHAR
   TYPE     17-24 CHAR
   DATE     41-46 CHAR
   TIME     47-52 CHAR
   RUNTIME  61-64 FIXED
   CPUTIME  65-68 FIXED
   LPAGBUFF 69-72 FIXED
   CENTURY  73-74 CHAR  NULL IF POS(73-74) = 0
 DATALOAD TABLE(SYSDETAIL) IF POS(75-78)='OSQL'
   SQLNAME   1-8  CHAR
   TYPE     17-24 CHAR
   DATE     41-46 CHAR
   TIME     47-52 CHAR
   RUNTIME  61-64 FIXED
   CPUTIME  65-68 FIXED
   LPAGBUFF 69-72 FIXED
   CENTURY  73-74 CHAR  NULL IF POS(73-74) = 0
 DATALOAD TABLE(SYSDETAIL) IF POS(75-78)='CSQL'
   SQLNAME   1-8  CHAR
   TYPE     17-24 CHAR
   DATE     41-46 CHAR
   TIME     47-52 CHAR
   RUNTIME  61-64 FIXED
   CPUTIME  65-68 FIXED
   LPAGBUFF 69-72 FIXED
   CENTURY  73-74 CHAR  NULL IF POS(73-74) = 0
```

Figure 82. Example DBS Utility Commands to Load Accounting Tables (Part 1 of 2)

```
 DATALOAD TABLE(USERDETAIL) IF POS(75-78)='USQL'
   SQLNAME   1-8  CHAR
   USERPART  9-16 CHAR
   SQLUSER  17-24 CHAR
   USERDATA 25-40 CHAR
   DATE     41-46 CHAR
   TIME     47-52 CHAR
   PNAME    53-60 CHAR
   ATIME    61-64 FIXED
   CPUTIME  65-68 FIXED
   ULPAGBUF 69-72 FIXED
   CENTURY  73-74 CHAR  NULL IF POS(73-74) = 0
 DATALOAD TABLE (DRDADETAIL) IF POS (75-78) = 'RSQL'
   SQLNAME     1-8      CHAR
   ACCUSRID    9-16     CHAR
   SQLUSER     17-24    CHAR
   DATE        25-30    CHAR
   TIME        31-36    CHAR
   LUWID       37-63    CHAR
   CENTURY     73-74    CHAR  NULL IF POS(73-74) = 0
INFILE(ARIACC1 RECFM(FB) RECSZ(80) BLKSZ(2000) PDEV(DASD))
COMMIT WORK;
SELECT * FROM SQLDETAIL;
SELECT * FROM SYSDETAIL;
SELECT * FROM USERDETAIL;
```

Figure 82. Example DBS Utility Commands to Load Accounting Tables (Part 2 of 2)

Converting VSAM ESDS Accounting File Records into VSAM Managed SAM Feature Records

If VSE/VSAM ESDS files are used to collect your accounting records, you will have to convert the records into VSAM managed SAM feature records before you load them into the tables. This is because for the DBSU DATALOAD function, only a local SAM file or VSAM managed SAM file can be processed as a DASD input file. Figure 83 shows sample JCL statements for converting VSAM ESDS accounting file records into VSAM managed SAM file records.

```
// DLBL ARIACT1,'ACCTFIL1',0,VSAM,CAT=SQLWK1,DISP=(OLD,KEEP)
// DLBL ARIACT2,'ACCTFIL2',0,VSAM,CAT=SQLWK1,DISP=(OLD,KEEP)
// DLBL ARIACC1,'ACCT.SAM.FILE1',0,VSAM,CAT=SQLWK1,               C
          RECSIZE=80,RECORDS=(X,Y),DISP=(NEW,KEEP)
// DLBL ARIACC2,'ACCT.SAM.FILE2',0,VSAM,CAT=SQLWK1,               C
          RECSIZE=80,RECORDS=(X,Y),DISP=(NEW,KEEP)
// EXEC IDCAMS,SIZE=AUTO
   REPRO INFILE(ARIACT1) -
   OUTFILE(ARIACC1 ENV(RECFM(FB) BLKSZ(2000) RECSZ(80))


   REPRO INFILE(ARIACT2) -
   OUTFILE(ARIACC2 ENV(RECFM(FB) BLKSZ(2000) RECSZ(80))

/*
```

Figure 83. Job Control for Converting VSAM ESDS File Records to VSAM Managed SAM File Records

Note:

1. ′ACCTFIL1′ and ′ACCTFIL2′ are the file ids of the VSAM ESDS accounting files used to collect accounting records.

2. ARIACC1 and ARIACC2 should be specified as the file name of the VSAM managed SAM file to be used as the IDCAMS REPRO command output files. After the conversion, by using the same set of DLBL statements, these files can be used as input files to the DBSU DATALOAD job control.
3. Set RECSZ to 80 and BLKSZ to 2000, they are the expected values for DBSU DATALOAD.
4. The example assumes that the VSAM managed SAM files ARIACC1 and ARIACC2 are implicitly defined to VSAM the first time they are opened.
5. DISP=(NEW,KEEP) indicates that the files will be reset at OPEN time. If there are still existing records in either ARIACC1 or ARIACC2 that are not yet processed, DATALOAD the records first before any new conversion.

Chapter 11. Generating Additional Databases

Initially, you set up database partition with one database: then, depending on your needs, you can add additional databases or database partitions. Your initial database is generated at the time of installation; you generate additional ones later. This chapter describes how to generate a database. It assumes that you are familiar with the terminology discussed in Figure 5 on page 7 and have reviewed Chapter 2, "Planning for Database Generation," on page 13. The database is specified when application server is started. This is done through the use of job control statements that reference the required database.

Learning about Configuration Concepts

Reasons for Adding a Database Partition

Initially, there is one database partition which is called SQLDS. As your installation grows, you can add more database partitions. The primary reason for doing so would be to permit multiple user mode access to more than one database at the same time, or *multiple database operation*.

Consider, for example, an installation having one database partition (SQLDS) and three databases (SQLDBA, DATA1, and DATA2). A database partition can manage only one database at a time. Thus, as Figure 84 on page 212 shows, while the database partition is accessing one database in multiple user mode, the other databases are inactive.

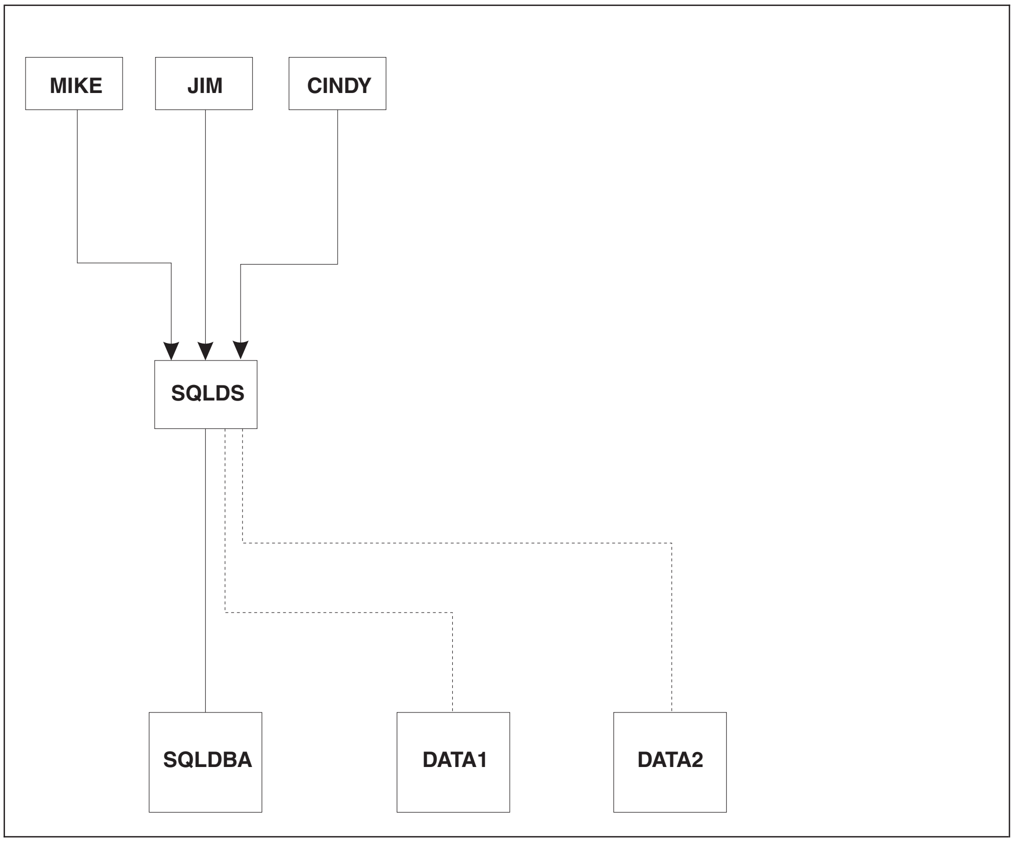


Figure 84. One Database Partition Accessing One Database

Users could access the remaining databases (DATA1 and DATA2) in single user mode if their partitions are properly prepared; however, it is not recommended that the database manager be used this way.

If you define two more database partitions, multiple user access to all three databases is possible at the same time. Figure 85 on page 213 shows a multiple database configuration. In this case, two more database partitions are defined (SQLMFB and SQLJDS). Each partition owns one database, and operates independently of the others.

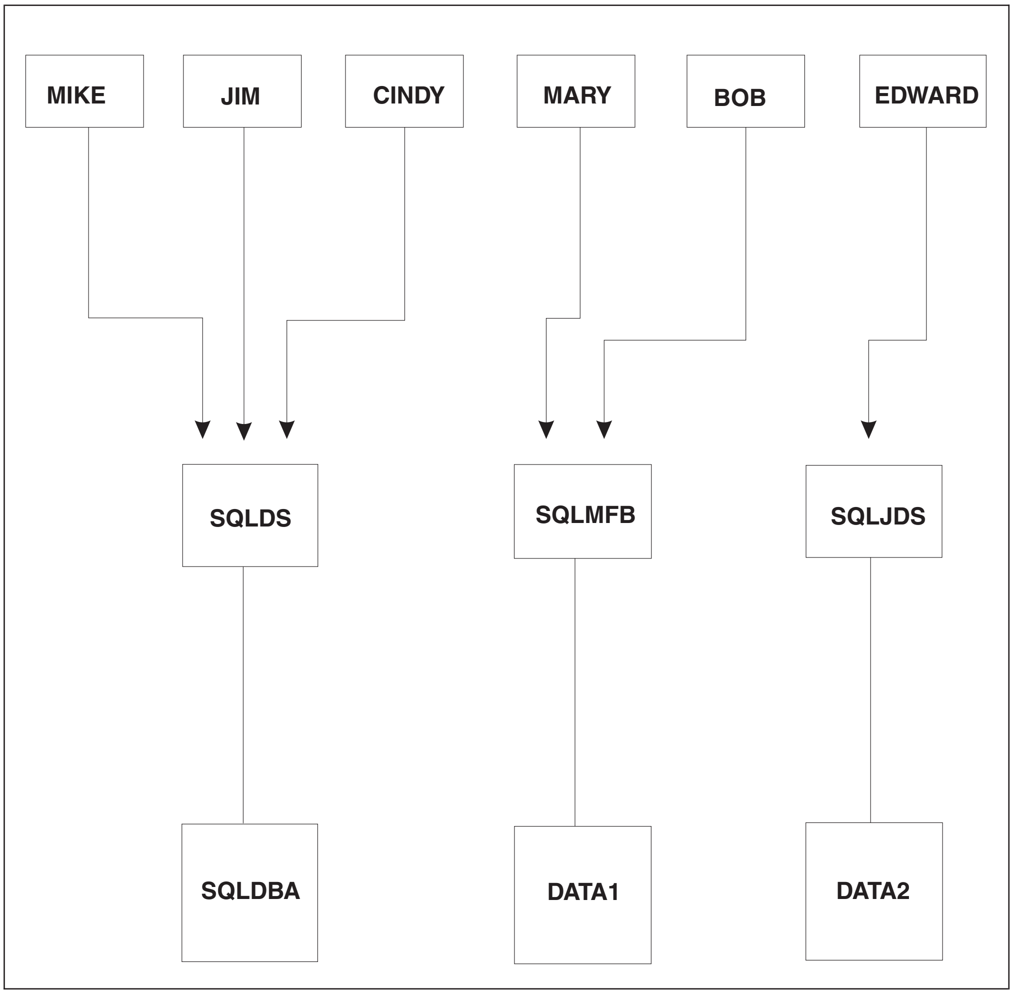


Figure 85. Multiple Users Accessing Multiple Databases

Database Generation Process

The steps for generating a database are as follows:

1. Update the DBNAME Directory for the new database.
2. Define the VSAM data sets for the database, by running the VSAM IDCAMS program with the appropriate set of DEFINE commands.
3. Set up the job control statements for generating the database.
4. Modify and run a job control to generate a database. The job control does the following:
   - Formats the database components and constructs the catalog tables
   - Installs the DBS utility
   - Runs the DBS utility to complete the installation of the database.

     This involves tasks such as creating views, granting access to DB2 Server for VSE facilities, and acquiring dbspaces for system use.

   Once the database is generated, you can do the following:
5. Install the desired components into the database, such as ISQL, online support, and HELP text.
6. Optionally change the application server default CHARNAME.
7. Optionally change the application server default character subtype.

8. Optionally set the DBCS option to YES.
9. Change the password of authorization ID SQLDBA in the database to one of your own choosing.
10. Optionally install the DRDA code.
11. Optionally load phases into the SVA.

These steps are all described in detail below.

Step 1: Update the DBNAME Directory

Update your DBNAME directory to add the new database. See "Setting Up the DBNAME Directory" on page 23.

Step 2: Defining the Database Data Sets

To define the VSAM data sets for the new database, run the VSAM utility program IDCAMS. The specific DEFINEs will depend on your requirements. At a minimum, however, you must define data sets for:
- A directory (sometimes called a BDISK)
- One, two, or four logs
- At least one dbextent (at least one data set).

The ARIS75CD procedure shown in Figure 86 on page 215 provides an example of the job control statements to define a VSAM user catalog and data sets for a database. This database has one directory, one log, and one dbextent. In the example, they will reside on an IBM 3380 DASD device.

Note: The VSAM keywords shown here are only the basic ones that the database manager requires. Other keywords and options that you can use are described in the *Using VSE/VSAM Commands and Macros* manual.

```
// JOB ARIS75CD              DB2 for VSE STARTER DATABASE VSAM DEFINITIONS
// LIBDEF PROC,SEARCH=(PRD2.DB2730)
// EXEC PROC=ARIS75DB        *-- SQL/DS DATABASE ID PROC
// EXEC IDCAMS,SIZE=AUTO
   DEFINE UCAT         /* DEFINE USER CATALOG      */ -
          ( NAME (SQLCAT)       -
            CYL (1)             -
            ORIGIN (NNNN)       -
            VOL (XXXXXX)      )
   DEFINE SPACE        /* DEFINE DB2    DATABASE SPACE */ -
          ( ORIGIN (NNNN)       -
            CYL    (119)        -
            VOL    (XXXXXX)   ) -
            CAT    (SQLCAT)
   DEFINE CLUSTER      /* DEFINE DB2    DATABASE DIRECTORY */ -
          ( NAME  (SQL.BDISK.STARTER.DB) -
            CNVSZ (512)         -
            CYL   (34)          -
            NONINDEXED          -
            VOL   (XXXXXX)      -
            RECSZ (505 505)     -
            REUSE               -
            SHR   (2)         ) -
            CAT   (SQLCAT)
   DEFINE CLUSTER      /* DEFINE DB2    DATABASE LOG */ -
          ( NAME  (SQL.LOGDSK1.STARTER.DB) -
            CNVSZ (4096)        -
            CYL   (08)          -
            NONINDEXED          -
            VOL   (XXXXXX)      -
            RECSZ (4089 4089)   -
            REUSE               -
            SHR   (2)         ) -
            CAT   (SQLCAT)
   DEFINE CLUSTER      /* DEFINE DB2    DATABASE DATA EXTENT 1  */ -
          ( NAME  (SQL.DDSK1.STARTER.DB) -
            CNVSZ (4096)        -
            CYL   (77)          -
            NONINDEXED          -
            VOL   (XXXXXX)      -
            RECSZ (4089 4089)   -
            REUSE               -
            SHR   (2)         ) -
            CAT   (SQLCAT)
/*
/&
```

Figure 86. Job ARIS75CD (Defining VSAM Data Sets for the Database)

Notes:

1 Change the NAME keyword to the name of the application server you want to access.

2 Change all occurrences of VOLUME (*XXXXXX*) and ORIGIN (*NNNN*) to reflect the volume serial number and origin allocation for the VSAM components that make up your new database. (The origin value is the beginning track number or block number.)

3 If you are allocating the database to a 3380 DASD device, you do not need to modify the CYL space allocations. If a fixed-block architecture (FBA) 3370/9332/9335 DASD device is being used for the database, replace the CYL allocations with the equivalent BLOCK allocations. For all other types of DASD devices, refer to Appendix B, “Estimating Database Storage,” on page 341. Minimum space allocation values are shown in Table 41 on page 342.

4 If the user catalog for the database will *not* be identified by the file-id SQLCAT, then:

- Alter the *file_id* in the DLBL statement identifying the VSAM user catalog for the database, and
- Alter the CAT specification in the IDCAMS DEFINE SPACE and DEFINE CLUSTER commands.

We recommend that you use a VSAM user catalog for the new database. If you choose not to, you must do the following:

- Remove the DLBL statement identifying the VSAM user catalog
- Remove the DEFINE UCAT command and the DEFINE SPACE command
- Remove the CAT specification from the IDCAMS DEFINE CLUSTER commands.

You may also wish to password-protect your database with the VSAM password protection facility. See “Protecting VSAM Data Sets” on page 119.

The directory for the database is defined as part of a VSAM services job that creates the initial set of database data sets. Figure 87 shows another example of defining a directory data set. Figure 86 on page 215 shows the complete job.

```
DEFINE CLUSTER
       (NAME (SQL.BDISK.DBNAME01.DB) -
        CNVSZ (512) -
        CYL (6) -
        NONINDEXED -
        VOL (volid1) -
        RECSZ (505 505) -
        REUSE -
        SHR (2) ) -
        CAT (SQLCAT01)
```

Figure 87. Defining the Directory Data Set

Notes:

1. The directory data set is defined by the DEFINE CLUSTER command. You can give it any name you like. To avoid confusion, however, you should retain BDISK as part of the name; then give it a qualifier to distinguish it from directories on other databases.
2. You must use the CNVSZ and RECSZ values shown. The directory must have 512-byte control intervals.
3. Set the directory size based on potential database size. Specify it with either CYL(*nn*) or TRK(*nn*) for count-key-data devices, or with BLOCKS(*nnnn*) for fixed block devices.
4. The SHR(2) parameter must be used to allow archiving.
5. You may wish to password-protect your database with the VSAM password protection facility. See “Protecting VSAM Data Sets” on page 119.

Step 3: Setting Up Your Database Job Control

Each time the database manager (program ARISQLDS) is run, the following job controls are needed:

- DLBL statements, which identify the database being accessed (the directory, the log or logs, and the data dbextents)
- LIBDEF statements, which define the DB2 Server for VM libraries and any other needed libraries
- Trace facility statements
- Database archive and log archive statements
- Accounting facility statements.

During the initial installation of the database manager, two procedures are defined that contain the needed LIBDEF statements:

ARIS75PL	runs LIBDEF statements to define the production libraries, which are required for the day-to-day use of the database manager.
ARIS75SL	runs LIBDEF statements to define libraries that are required for database generation and for code link edits.

It is recommended that you also use catalogued procedures for the other job controls, to avoid having to maintain multiple copies of them. Because procedures for the libraries are cataloged during installation, you only need to catalog the DLBL statements that identify the new database. You also should catalog the job control statements needed for the trace, database archive, log archive, and accounting facilities.

Database archiving and log archiving can be directed only to tape, so for these activities, you must use TLBL statements. Trace output and accounting output can be directed to either DASD or tape. For information on specifying job control statements for the trace output file, see the *DB2 Server for VSE & VM Operation* manual; for information on accounting job control statements, see “Setting Up a Job Control for the Accounting Files” on page 187.

Figure 88 on page 218 shows an example of the job control statements needed for cataloging the database.

```
// JOB CATALOG DATABASE JOB CONTROL
// EXEC LIBR
ACCESS SUBLIB=PRD2.DB2730
CATALOG DBNAME01.PROC
// DLBL IJSYSUC,'SQLCAT01',,VSAM
// DLBL BDISK,'SQL.BDISK.DBNAME01.DB',,VSAM
// DLBL LOGDSK1,'SQL.LOGDSK1.DBNAME01.DB',,VSAM
// DLBL LOGDSK2,'SQL.LOGDSK2.DBNAME01.DB',,VSAM
// DLBL ALTLGD1,'SQL.ALTLGD1.DBNAME01.DB',,VSAM
// DLBL ALTLGD2,'SQL.ALTLGD2.DBNAME01.DB',,VSAM
// DLBL DDSK1,'SQL.DDSK1.DBNAME01.DB',,VSAM
// DLBL DDSK2,'SQL.DDSK2.DBNAME01.DB',,VSAM
// TLBL ARITRAC,...
// TLBL ARIARCH,...
// TLBL ARILARC,...
// TLBL ARILALT,...
// DLBL ARIACC1,...
// EXTENT...
// DLBL ARIACC2,...
// EXTENT...
/+
/*
/&
```

Figure 88. Job for Cataloging Database Job Control

Notes:

1. Specify the sublibrary into which your procedure is to be cataloged. In the above example, PRD2.DB2730 is used.
2. The file names on the DLBL and TLBL statements must be as shown in the example.
3. The data set names on the DLBL and TLBL statements must match those on the DEFINE CLUSTER commands that defined the data sets for the database.
4. You need a DLBL statement for the directory (BDISK).
5. You need a DLBL statement for each log data set (LOGDISK1, LOGDISK2, ALTLGD1, ALTLGD2).
6. You need one DLBL statement for each defined dbextent data set (DDSK1, DDSK2).
7. If you plan to use tracing, you need job control for the trace output file. The file name must be ARITRAC. This example shows a TLBL statement; trace output can be directed to disk as well.
8. If you will be running with LOGMODE=A or L, you need a TLBL statement for archiving the database. The file name must be ARIARCH.
9. If you will be running with LOGMODE=L, you also need a TLBL statement for archiving the log. The file name must be ARILARC. If you use alternate logging, you also need a TLBL for the inactive log. The file name must be ARILALT.

   It is recommended that you do not specify any VOLID parameter on the TLBL statements for log archiving. Because multiple files can be created or read during the same run of the database manager, you would want different VOLIDs for the different files.
10. If you are including job control statements for the accounting file facility, the file name of the first accounting file must be ARIACC1, and that of the second file, if you choose to have it, must be ARIACC2. Only one file is required, but it is recommended that you use two.

Step 4: Generating the Database

To generate a database, modify and run the job control shown in Figure 89 in single user mode (SYSMODE=S). This will run IBM-supplied procedures to do the following:

1. Create the package for the DBS utility using the Assembler preprocessor
2. Finish the database generation process using the DBS utility.

```
// JOB GENERATE A DATABASE NAMED DBNAME01
// EXEC PROC=ARIS75SL    *-- SERVICE/PRODUCTION LIBRARY ID PROC
// EXEC PROC=DBNAME01    *-- DATABASE ID PROC
// EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,STARTUP=C'
CUREXTNT=2
MAXEXTNT=200                   The meanings of these
MAXDBSPC=1000                  parameters are described
END                            in the paragraphs that
POOL 2 NOLOG                   follow this figure.
1 1
2 2
END
PUBLIC  12800  1
PUBLIC   2048  1
PUBLIC   8192  1
PUBLIC   1024  1
PUBLIC    512  1
PUBLIC    512  1
PUBLIC    512  1
PUBLIC    512  2
PUBLIC    512  2
PRIVATE   128  1
PRIVATE   128  1
PRIVATE   512  1
PRIVATE   128  2
PRIVATE   128  2
PRIVATE   512  2
INTERNAL   80  1024 1
END
/*
// EXEC PROC=ARIS040D    *-- PREP DBS UTILITY
// EXEC PROC=ARIS050D    *-- PERFORM REQUIRED DATABASE SET UP
// EXEC PROC=ARISDBSD    *-- LOAD SAMPLE TABLES AND ROUTINES
READ MEMBER ARISAMDB
READ MEMBER ARISAMPI
/*
/&
```

Figure 89. Example of a Job Control for Generating Your Own Database

Notes:

1. The ARIS75SL procedure is cataloged during initial installation. It contains job control statements that identify the appropriate set of service libraries.
2. The DBNAME01 cataloged procedure refers to the job control (DLBL and LIBDEF) statements for the database being generated. You define and create this procedure as explained in “Step 3: Setting Up Your Database Job Control” on page 217.
3. Running the database manager (PGM=ARISQLDS) in single user mode (SYSMODE=S) with STARTUP=C calls the database generation program. This program reads the SYSIPT input control cards that specify how the database is to be generated. These statements are described later. Program ARISQLDS also

reads the A-type source member ARISCAT. The contents of this member remain constant for all databases and should not be changed.

4. The ARIS040D procedure should be used without modification. It preprocesses the DBS utility, and references DB2 Server for VSE source members.
5. Procedure ARIS050D uses the DBS utility to process the SQL statements that finish generating the database. Do not change this procedure or the A-type source member it reads (ARISDBU), because you may want to use them to install and generate additional databases in the future. If you want to allocate larger dbspaces for the HELP text, ISQL-stored queries, or sample tables than those defined in ARISDBU, copy and rename both ARIS050D and ARISDBU; then increase the number of pages in the ACQUIRE PUBLIC DBSPACE statements in the renamed member, and update the renamed procedure to use the renamed member as input.

   The member ARISDBU enables you to perform the following:
   - Grant RUN authority to PUBLIC for the DBS utility package SQLDBA.ARIDSQL
   - Acquire the standard database public dbspaces for HELPTEXT, ISQL, and SAMPLE
   - Create the HELP text tables SQLDBA.SYSTEXT1, SQLDBA.SYSTEXT2, and SQLDBA.SYSLANGUAGE
   - Create indexes SQLDBA.SYSTEXT1INDEX, SQLDBA.SYSTEXT2INDEX, and SQLDBA.SYSLANGINDEX on the HELP text tables
   - Create the ISQL sample routine table EXAMPLE.ROUTINE and its index EXAMPLE.RINDEX
   - Create the SYSUSERLIST catalog view and grant access to PUBLIC
   - Grant access to PUBLIC on all catalog tables except SYSTEM.SYSUSERAUTH.

   Note: You should not drop the dbspaces or tables that are acquired and created above, even if you are not installing the corresponding facilities. Certain database maintenance operations assume that these tables and dbspaces exist in the database.
6. The procedure ARISDBSD runs the DBS utility in single user mode.
7. The A-type source members ARISAMDB and ARISAMPI contain SQL statements to be run by the DBS utility to build and load the sample tables and routines.

The input (SYSIPT) control statements for the database generation program are divided into three sets of input records, separated by END delimiter control statements. These specify:

- *Database generation keyword control statements*, which define the number of dbextents to be prepared during the database generation process (CUREXTNT), and establish certain maximum values for the database (MAXPOOLS, MAXEXTNT, and MAXDBSPC). Each statement can be specified on its own input record (card), or multiple statements can be specified on one input record. An END delimiter control statement must be specified after all the keyword control statements. The CUREXTNT control statement must be specified; all the others have default values.
- *Initial storage pool and dbextent definitions*, which identify the initial set of nonrecoverable storage pools and define the initial set of dbextents. Any dbextent defined here must have a corresponding DLBL statement in your database job control (DBNAME01). You must specify at least one dbextent for

each storage pool that is referenced by the initial dbspace definitions. The POOL control statements that define nonrecoverable storage pools must precede the statements that define the dbextents.

- *Initial dbspace definitions*, which define the initial set of dbspaces, including public dbspaces required by the database manager (system dbspaces), any user public and private dbspaces you need initially, and the internal dbspace allocations for the database. You must specify at least five public dbspaces: first two for the database catalog and package storage, and three more for the HELP text, ISQL tables, and the sample tables. In Figure 89 on page 219 the first five dbspace keyword control statements define these five dbspaces. The remainder of the public and private dbspaces shown in Figure 89 on page 219 are user dbspaces of various sizes and storage pool assignments.

  You must also specify the internal dbspaces for the database. You can change the specification of internal dbspaces on any ADD DBSPACE operation. For information, see “Adding Dbspaces to the Database” on page 125.

The details of specifying these database generation control statements are described below.

Specifying Keyword Control Statements

The format for specifying the database generation keyword control statements is:

```
►►──CUREXTNT=nnn──┬──────────────┬──┬──────────────┬──┬──────────────┬──END──────────────►◄
                  └─MAXPOOLS=nnn─┘  └─MAXEXTNT=nnn─┘  └─MAXDBSPC=nnn─┘
```

CUREXTNT

CUREXTNT specifies the number of dbextents being defined in the database generation. You must specify it; it has no default value. Its value can be from 1 to 999, and must match the number of dbextent definition control statements. You must also have DLBL statements for all the dbextents being defined (in your DBNAME01 procedure). In the example shown in Figure 89 on page 219, CUREXTNT=2 indicates that two dbextents are being defined.

MAXPOOLS

MAXPOOLS specifies the maximum number of storage pools that can ever be defined for the database. Its value can range from 1 to 999. The default is 32. In the example in Figure 89 on page 219, the default is used.

MAXEXTNT

MAXEXTNT specifies the maximum number of dbextents that can ever be defined for the database. Its value can range from 1 to 999. The default is 64. In the example in Figure 89 on page 219, it is set to 200.

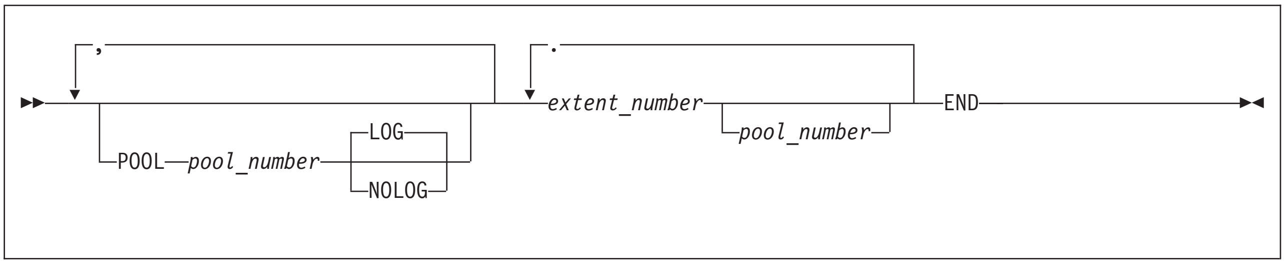
MAXDBSPC

MAXDBSPC specifies the maximum number of dbspaces that can ever be defined for the database. Its value can range from the number you specify in your database generation control statements to 32000. The default is 1000. In the example in Figure 89 on page 219, it is explicitly set to 1000.

The keyword control statements must be coded in columns 1-71. Column 72 is a continuation column, and columns 73-80 are ignored. If you specify more than one keyword control statement on a single input record, separate them with blanks.

Specifying Initial Storage Pools and Dbextents

The control statement format for specifying the initial storage pools and dbextents is:



POOL

Include the POOL control statement only for those storage pools you want to define as nonrecoverable: if you omit it, the pool will be defined as recoverable. You can specify as many nonrecoverable storage pools as you want, up to the MAXPOOLS value. For more information, see “Nonrecoverable Storage Pools” on page 177.

pool_number

The value for *pool_number* is the number of the storage pool. You cannot specify 1 because storage pool 1 is the default storage pool for dbspaces, so it cannot be defined as nonrecoverable.

LOG

The LOG option, which indicates that the storage pool is to be recoverable, is the default. Specify the NOLOG option if the storage pool is to be nonrecoverable.

extent_number/pool_number

The dbextent definition control statements follow the POOL statements. They define an initial set of dbextents (by number), and the storage pool assignment for each.

The first number in the pair is the extent number, which corresponds to the suffix number in the file name of the dbextent data sets (DDSK*n*). You must use stylized file names on the DLBL statements describing the dbextents. The file name is DDSK*n*, where *n* is the extent number used in the dbextent definition record. You must define the dbextents in consecutive (numeric) order by extent number.

The second number, which must be separated from the first by at least one blank, is the storage pool number. If you do not specify the storage pool number, it defaults to 1.

Note: You cannot assign a dbspace to a storage pool until a dbextent has been assigned to it.

Each extent number/storage pool number pair must be entered on a separate input record. You can put comments on the dbextent control statements, by specifying the storage pool number and separating the comment from the number by at least one blank. A comment must be contained in the one input record for the dbextent: it cannot be continued on the next input record, which is interpreted as the next dbextent definition.

In the example in Figure 89 on page 219, dbextent number 1 (DDSK1) is assigned to storage pool number 1, and dbextent number 2 (DDSK2) is assigned to storage pool number 2.

Specifying Initial Dbspaces

The format is:

```
                                           ┌─1───────────────────┐
►►──┬─PUBLIC───────┬─number_of_pages───────┼─────────────────────┼──────────────────────────────►◄
    │              │                       └─storage_pool_number─┘
    └──┬─PRIVATE─┬─┘
```

The *number_of_pages* value is the number of logical pages in the dbspace, rounded up to the next higher multiple of 128. The *storage_pool_number* value must correspond to a pool that already has a dbextent defined for it, as defined by the dbextent control statements.

You must define five public dbspaces for system use: the catalog dbspace, package, HELP text, ISQL tables, and sample tables dbspaces. In the example shown in Figure 89 on page 219, all are assigned to storage pool 1, but you can assign them elsewhere. The catalog and package dbspaces must always be assigned to a recoverable storage pool. In the example in Figure 89 on page 219, the first five dbspace control statements specify:

- 12800 pages for the catalog dbspace (SYS0001)
- 2048 pages for the package dbspace (SYS0002)
- 8192 pages for the HELPTEXT dbspace
- 1024 pages for the ISQL dbspace
- 512 pages for the SAMPLE dbspace.

The general format for specifying the initial internal dbspace control statement is:

```
                                                    ┌─1───────────────────┐
►►──INTERNAL──number_of_dbspaces──number_of_pages───┼─────────────────────┼──────────────►◄
                                                    └─storage_pool_number─┘
```

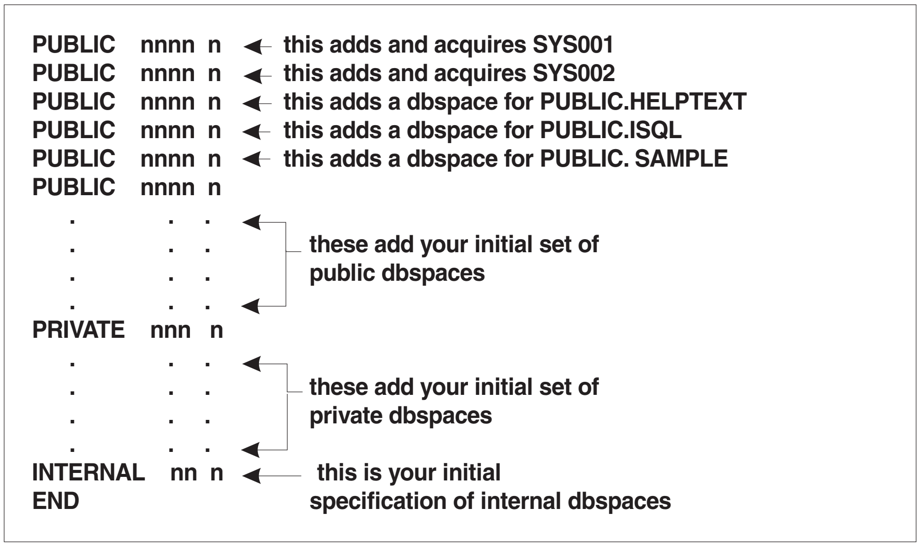
This statement specifies the number (*number_of_dbspaces*) of equal size (*number_of_pages*) temporary dbspaces that the database manager can use for internal sorting and index creation. The *storage_pool_number* must correspond to a pool that already has a dbextent in it, as defined by the dbextent control statements. You must not delete the last dbextent from the storage pool that contains the internal dbspaces. The storage pool to which you assign the internal dbspaces can be either recoverable or nonrecoverable: if you do not specify the storage pool number, it defaults to 1.

This internal dbspace keyword control statement must be the last dbspace definition input record before the END delimiter control statement. Separate the values you specify in this statement by at least one blank.

In the example in Figure 89 on page 219, 80 internal dbspaces of 1024 pages each are defined and assigned to storage pool 1.

Note: Because the catalog and package dbspaces are assigned to storage pool 1, performance is improved if you assign the internal dbspaces to some other recoverable storage pool. In this example, they are assigned to storage pool 1, just to keep things simple.

Generally speaking, your input records for initial dbspace definitions would follow this pattern:



```
PUBLIC   nnnn n  <-- this adds and acquires SYS001
PUBLIC   nnnn n  <-- this adds and acquires SYS002
PUBLIC   nnnn n  <-- this adds a dbspace for PUBLIC.HELPTEXT
PUBLIC   nnnn n  <-- this adds a dbspace for PUBLIC.ISQL
PUBLIC   nnnn n  <-- this adds a dbspace for PUBLIC. SAMPLE
PUBLIC   nnnn n
  .         . .  <--|
  .         . .     |-- these add your initial set of
  .         . .     |   public dbspaces
  .         . .  <--|
PRIVATE   nnn  n
  .         . .  <--|
  .         . .     |-- these add your initial set of
  .         . .     |   private dbspaces
  .         . .  <--|
INTERNAL   nn n  <----  this is your initial
END                     specification of internal dbspaces
```

Figure 90. Input Records for Initial Dbspace Definitions

The first two dbspaces are public dbspaces: PUBLIC.SYS0001 and PUBLIC.SYS0002, which are both defined and acquired by the generation process for the catalog tables and for the packages, respectively. You are advised to change these control statements only if you want to define and allocate a larger dbspace for the catalog tables or for the package dbspace, respectively. To do this, increase the *number_of_pages* value in the control statement.

The third, fourth, and fifth dbspaces are public dbspaces that are added by the generation process. They are later acquired when procedure ARIS050D calls the DBS utility to complete the generation of the database. As shown in Figure 89 on page 219, procedure ARIS050D must be called whenever you generate a database.

Note: The fifth dbspace, PUBLIC.SAMPLE, is also added by the generation process. It is used to hold the IBM-supplied sample data tables. These tables are created and loaded during the last run of the DBS utility, when the A-type members ARISAMDB and ARISAMPI are processed. Details of ARISAMDB and ARISAMPI are provided in the *DB2 Server for VSE Program Directory* manual. The data in the sample tables is manipulated by sample application programs, which are called by the IBM-supplied job control members. There is one sample program for each programming language that the database manager supports. Details of these programs are in the *DB2 Server for VSE & VM Application Programming* manual.

You **must** specify database generation control statements for all five of these dbspaces. If you omit one, the database generation may fail.

Code the dbspace values in columns 1-71. Columns 72-80 are ignored. You can put comments on the dbspace statements by specifying the storage pool number and separating the comment from this number by at least one blank.

Step 5: Installing the Database Components

After the database is generated, you can install these three components into it:
- HELP text
- Online support
- ISQL.

All of these components are optional; which ones you should install depends on your usage environment. For example, in a query/report writing environment, you should install all three.

Regardless of whether you use these components, you can still install all of them into the database for possible future use. Figure 91 shows job control statements that install all the optional components into the DBNAME01 database. Modify the job control and run it to install the components into your database. (The database manager must be running in single user mode.)

```
// JOB TO INSTALL DATABASE COMPONENTS
// EXEC PROC=ARIS75SL   *--SERVICE/PRODUCTION LIBRARY ID PROC
// EXEC PROC=DBNAME01   *--DATABASE ID PROC
// EXEC PROC=ARIS380D,LANG=AME,HELP=ONLY,CUU=xxx
// EXEC PROC=ARIS080D   *--INSTALL ONLINE SUPPORT
// EXEC PROC=ARIS110D   *--INSTALL ISQL
// EXEC PROC=ARIS120D   *--INSTALL ISQL
// EXEC PROC=ARIS130D   *--INSTALL ISQL
/&
```

Figure 91. Job Control to Install Optional Database Components

Notes:

1. Procedure ARIS380D loads the English version of the DB2 Server for VSE HELP text into the database. About 40000 rows are inserted, and the job normally takes 10 to 15 minutes. The dbspace PUBLIC.HELPTEXT must exist for this procedure to be run. If you do not want the English version of the HELP text, you can omit the job control statement for this procedure.

   HELP text is loaded from the Help Text Tape, which must be mounted on the tape drive identified by the value assigned to the cuu parameter.

   HELP text is available in other languages as well. To load HELP text for a language other than American English, replace the value for the LANG parameter in the PROC ARIS380D with one of the following values.

AMENG	American English
UCENG	Uppercase English
FRANC	French
GER	German

KANJI	Kanji (Japanese)
HANZI	Simplified Chinese

Refer to the installation instructions in the *DB2 Server for VSE Program Directory* manual.

2. Procedure ARIS080D installs the online support into the database, and grants CONNECT authority to ALLUSERS. This allows the online support to implicitly connect users. For information on implicit CONNECT, see the *DB2 Server for VSE & VM Database Administration* manual.

   If the online support is not required for your database, omit the job control statements for procedures ARIS080D, ARIS110D, ARIS120D, and ARIS130D.
3. Procedures ARIS110D, ARIS120D, and ARIS130D install ISQL support into the database. If you do not want to install the ISQL support, omit the job control statements for these procedures. You may also omit the job control statement for procedure ARIS060D.

After making the necessary job control modifications, submit the job for processing. All steps should end with a return code of 0 or 4. If any step fails to run, remove the EXEC PROC statements for all job steps that completed, and rerun the job. However, do not remove the database identification procedure or the library definition procedure (DBNAME01 and ARIS75SL). For example, if the step EXEC PROC=ARIS130D did not complete successfully, you can rerun the step by running the job control shown in Figure 92:

```
// JOB RESTART
// EXEC PROC=DBNAME01   *--YOUR DATABASE IDENTIFICATION PROCEDURE
// EXEC PROC=ARIS75SL   *--SERVICE/PRODUCTION LIBRARY ID
// EXEC PROC=ARIS130D   *--INSTALL ISQL
/&
```

Figure 92. Job Restart for Installing Optional Components

Step 6: Reload CCSID-Related Packages

The database manager and online resource manager use CCSID-related phases for validating and folding characters in SQL statements. The programs used to create these phases depend on packages residing in the database. The job control in Figure 93 can be used to load the package in the new database.

```
// JOB RELOAD CCSID-RELATED PHASES PACKAGE
// EXEC PROC=ARIS75SL   *--SERVICE/PRODUCTION LIBRARY ID
// EXEC PROC=DBNAME01   *--YOUR DATABASE IDENTIFICATION PROCEDURE
// EXEC PROC=ARIS175D   *--RELOAD CCSID-RELATED PHASES PACKAGE
/&
```

Figure 93. Job Reload for Loading CCSID-Related Phases Package

Step 7: Optionally Changing the Application Server Default CHARNAME

The application server default CHARNAME value on a newly installed database manager is INTERNATIONAL (CCSID=500). On a migrated database manager, the default is ENGLISH (CCSID=37). To change the application server default

CHARNAME (and with it the application server default CCSID, classification tables, and translation tables), specify the new CHARNAME initialization parameter.

For information on creating a new CHARNAME, CCSID, and character set, see Chapter 12, "Choosing a National Language and Defining Character Sets," on page 231.

Step 8: Optionally Changing the Application Server Default Character Subtype

If you use mixed character data (which contains DBCS and SBCS characters), you may want to change the application server default character subtype (CHARSUB) to mixed. The application server default character subtype is the value used for new columns when the character subtype is not explicitly defined by the CREATE TABLE or ALTER TABLE statements, or supplied as a package option. The character subtype value is also used to determine whether the results of the CHAR, DIGITS, and HEX scalar functions and the character representation of date, time, or timestamp values, or special registers should be interpreted either as mixed data or as SBCS data.

The application server default character subtype is initially set to SBCS.

For information on changing the default character subtype, see "Setting the Application Server Default Character Subtype" on page 253.

Step 9: Optionally Setting the DBCS Option to YES

If you are using a double-byte character set (DBCS), you should enable the DBCS option, which allows the database manager to correctly interpret SQL statements that contain DBCS strings. As a default, the DBCS option is not enabled. For information, see "Using Double-Byte Character Set (DBCS)" on page 243.

Step 10: Changing the Password of Authorization ID SQLDBA

One final task you should not omit is to change the password for the authorization ID SQLDBA in your new database. The authorization ID SQLDBA is defined in all databases to have DBA authority. The password is set to SQLDBAPW during database generation. Because this default password for SQLDBA is common knowledge (it is in many product manuals), you should change it immediately after database generation. To do so, use ISQL, an application program, or the DBS utility to connect to the database manager as SQLDBA with the following statement:

```
CONNECT SQLDBA IDENTIFIED BY SQLDBAPW
```

Then change the password to one of your own choosing with the following statement:

```
GRANT CONNECT TO SQLDBA IDENTIFIED BY newpw
```

Step 11: Optionally Install the DRDA Code

A DRDA environment provides the application server capability for remote unit of work access to data that is distributed across different installations. For more information on installing this code, see Chapter 14, "Using a DRDA Environment," on page 313.

Step 12: Optionally Load Phases into SVA

The VSE shared virtual area (SVA) allows the DB2 Server for VSE code to be shared. This sharing can reduce the amount of storage required by the database partitions, and reduces the amount of paging done by the system. You can load the eligible phases into the shared virtual area.

Table 18 shows the phases that are eligible to be loaded into the shared virtual area.

Table 18. Phases Eligible for SVA

Component	Phase
DBSS	ARISQLDS
RDS	ARIXRDS
DBSU	ARIDBS
Batch Resource Adapter	ARIRBARM
DBNAME Directory	ARICDIRD
Assembler Preprocessor	ARIPRPA
COBOL Preprocessor	ARIPRPC
FORTRAN Preprocessor	ARIPRPF
PL/I Preprocessor	ARIPRPP

Notes:

1. If you load all of the eligible phases, allocate an additional 4.1 megabytes of storage to the shared virtual area.
2. The DBSS component contains the code for the following components:
   - DBSS
   - DSC
3. The RDS component contains the code for the following components:
   - RDS
   - WUM
   - DRRM
   - CONV

   The DRRM and WUM components are only applicable if the DRDA code is installed.
4. If loaded, the DBSS and RDS components must both be loaded together in the SVA.
5. Batch Resource Adapter phase or any other phase that uses socket macro calls, cannot be loaded in SVA while using CSI assembler interface for TCP/IP.

Phases can be loaded into the shared virtual area in the following ways:

1. During VSE system IPL, add the SET SDL command into the IPL PROC.
2. After IPL, a separate job control with the SET SDL command can be run anytime after IPL.

The SET SDL command must always be issued from the background partition. Once the phases have been loaded, they cannot be purged until the next VSE system IPL.

The following is a sample job control to load both the DBSS and RDS phase into the shared virtual area:

```
// JOB LOADSVA
// LIBDEF PHASE,SEARCH=(IJSYSRS.SYSLIB,PRD2.DB2730)
   SET SDL
   ARISQLDS,SVA
   ARIXRDS,SVA
/*
/&
```

For more information on the shared virtual area, see the *IBM VSE/ESA System Control Statements* manual.

Chapter 12. Choosing a National Language and Defining Character Sets

A national language (as opposed to a programming language) is a language used in or by a nation. The database manager can work with data in national languages represented by a single-byte character set (SBCS) or by a double-byte character set (DBCS). The database manager will also support MBCS data from other platforms, which will be converted to SBCS and DBCS data. The following are some of the single-byte character sets that are shipped with the database manager:

- French
- English
- Spanish
- Italian
- German.

Examples of double-byte character sets that are shipped with the database manager include:

- Japanese
- Chinese.

If you want a complete list of the character sets that are available, review the SYSTEM.SYSCHARSETS catalog table.

This chapter describes the facilities that the database manager provides for national languages:

- CHARNAME specification

  This facility allows you to specify character sets and CCSIDs other than the installation or migration default. The application server default CHARNAME for a new installation is INTERNATIONAL (CCSID=500). The application server default CHARNAME for a migrated system is ENGLISH (CCSID=37). The application requester default CHARNAME is always INTERNATIONAL (CCSID=500). The database manager can use alternative character sets for identifying character usage and for folding lowercase characters to uppercase.

  This facility provides for the proper interpretation and use of national language characters not included in the default character set, for example, characters with umlauts, accents, and tildes.

  This facility also provides for the proper interpretation of data from application requesters or application servers which use different character sets and code pages. Character conversion is performed on data when the CCSID of the application requester and the CCSID of the application server are different. The application server default CCSID is determined from the application server default CHARNAME.

  It is very important that the application server and application requester have the same CCSID value unless there is a specific reason for them to be different. When the application server and application requester have different CCSID values, character conversion cannot be avoided. This conversion has an associated performance overhead, and causes performance degradation. CCSID conversion of data also affects the sargability of predicates. For more information on performance, see the *DB2 Server for VSE & VM Performance Tuning Handbook* manual.
- DBCS option

This option, when it is set to YES, allows the database manager to correctly interpret the shift-out (X'0E') and shift-in (X'0F') characters that delimit EBCDIC DBCS strings. The DBCS option is set on both the application server and the application requester.

- Multiple language messages

  The database manager provides multiple language message support to allow users to select the language in which error and informational messages appear (the language must already be installed). The operator can select the language for operator messages.

  If an ISQL user selects a national language that is different from the national language set by the operator on the application server, when the ISQL user issues an operator command the output is in the language set by the operator.

  For example, the operator sets the application server national language for operator messages to ENGLISH. The ISQL users set the application requester to KANJI. When the user issues an operator command, the result is in ENGLISH.

- Multiple language HELP text.

  The database manager provides multiple language HELP text support. ISQL users can interactively retrieve help information on messages and codes and command reference information. The help facility allows ISQL users to retrieve help in the language of their choice (provided that the language version of the HELP text is installed). Information on DB2 Server for VSE HELP text is contained in this chapter, and in the *DB2 Server for VSE & VM Database Administration* manual.

The database manager also provides graphic data types for use with strings of DBCS characters, as well as a mixed subtype for character data that contains both DBCS and SBCS characters. For more information about using graphic and mixed data, refer to the *DB2 Server for VSE & VM SQL Reference* manual.

Considerations when changing default CHARNAME and CCSID

If you are not using the default CHARNAME or CCSID during installation or migration, ensure you consider the following activities:

1. Choosing default CHARNAME and CCSID for the application server
   - Installation - see "Choosing the Application Server Default CHARNAME and CCSID" on page 31.
   - Migration - see "Choosing an Application Server Default CHARNAME" on page 40.

   Note: Refer to "CCSID Conversion" on page 245 and "Determining CCSID Values" on page 248 for more information on CCSIDs.
2. Setting migration CCSID values
   - Installation - defaults are adequate.
   - Migration - see "Setting Migration CCSID Values" on page 41.
3. Optionally, choosing application server default character subtype
   - Installation and migration - see "Choosing the Application Server Default Character Subtype" on page 33.
4. Optionally, setting the DBCS option for the application server
   - Installation and migration - see "Setting the DBCS Option for the Application Server" on page 254.

Note: To understand the effect of DBCS options, refer to "Using Double-Byte Character Set (DBCS)" on page 243.

Changing from pre-Euro CHARNAME to Euro-compatible CHARNAME

DB2 Server for VSE & VM supports several code pages that are identical to existing code pages except that they include the Euro currency symbol rather than the International currency symbol (¤). If you choose to use a CHARNAME that corresponds to a CCSID for a code page that includes the Euro currency symbol, it is recommended that your existing character data that is currently tagged with a non-Euro CCSID be re-tagged with the corresponding Euro-compatible CCSID. These steps should only be used when changing your CCSID to the corresponding Euro-compatible CCSID as described in the following table:

Table 19. Non-Euro and Corresponding Euro-compatible CHARNAMEs and CCSIDs

From CCSID	From CHARNAME	To CCSID	To CHARNAME
37	English	1140	E-English
277	Danish-Norweigan	1142	EDanish-Norweigan
278	Finnish-Swedish	1143	EFinnish-Swedish
284	Spanish	1145	E-Spanish
285	UK-English	1146	E-UK-English
297	French	1147	E-French
500	International	1148	E-International
273	German	1141	E-German
280	Italian	1144	E-Italian

Step 1: This step will locate character data that currently represents the International currency symbol (¤). The Euro-compatible code pages have replaced the International currency symbol with the Euro symbol. If your database does not contain character data that represents the International currency symbol, you can skip this step. If your data does contain character data that represents the International currency symbol, you must decide how to handle this. You can either skip this step, in which case character data that currently represents the International currency symbol will be interpreted as the Euro symbol once the CCSID is changed, or, you can change the character data that currently represents the International currency symbol to some other value. The following SELECT statement will locate columns that are tagged with the non-Euro compatible CCSID.

```
SELECT CNAME,TNAME,CREATOR FROM SYSTEM.SYSCOLUMNS WHERE CCSID=current_ccsid
```

For each column found, run the following SELECT statement:

```
SELECT * FROM creater.tname WHERE cname LIKE'%cur_symbol%'
```

Note: If your keyboard cannot generate the International currency symbol, replace *cur_symbol* with the hex value X'9F'. In order to do this, you will have to use an editor that allows hexadecimal characters to be entered.

This command will show you all the rows for column *cname* that include the International currency symbol. You can now decide the appropriate value to change the International currency symbol to for each row.

Step 2: Re-tag existing character data with the new Euro-compatible CCSID. This step uses the JCL job ARISCSID, which is provided by DB2 Server for VSE & VM.

1. Shut down the database server.
2. If you use any database initialization parameters that must be specified when the database manager is started in single user mode, specify them in ARISCSID. Do **not** delete any parameters that are currently listed.
3. Make any changes to ARISCSID that are necessary for your local installation.
4. Submit ARISCSID for execution.
5. Start the database server using your normal procedures.

```
// JOB ARICCSID   UPDATE CCSID COLUMN OF SYSCOLUMNS
// LIBDEF PROC,SEARCH=(PRD2.DB2730)
// EXEC PROC=ARIS75PL     *-- DB2/VSE PROD. LIBRARY ID PROC
// EXEC PROC=ARIS75DB     *-- DB2/VSE DATABASE ID PROC
// EXEC ARISQLDS,SIZE=AUTO,PARM='SYSMODE=S,PROGNAME=ARIDBS,SERVAIDS=0000001'
COMMENT ' Do NOT alter the line above in any way ! '
COMMENT 'Replace "sqldbapw" with the password for the SQLDBA SQL user ID'
CONNECT SQLDBA IDENTIFIED BY sqldbapw;
COMMENT 'Replace "euro-ccsid" with the CCSID of the                 '
COMMENT 'Euro-compatible CHARNAME and replace "current_ccsid with'
COMMENT 'the CCSID of your current CHARNAME                         '
UPDATE SYSTEM.SYSCOLUMNS SET CCSID=euro-ccsid WHERE CCSID=current_ccsid;
COMMIT WORK;
/*
/&
```

Figure 94. ARICCSID - Sample JCL

Using Alternative Character Sets

When the database manager folds keywords and identifiers from lowercase to uppercase, or folds user-supplied data using the default TRANSLATE scalar function, it bases the folding on the default character set specified on the SQLSTART initialization parameter. For information on setting the default character set, refer to “Choosing an Application Server Default CHARNAME” on page 40.

Some characters in other national languages must be delimited by double quotation marks (") before they can be accepted in identifiers. The double quotation marks indicate that special characters are within the identifier. No characters within delimited identifiers are folded from lowercase to uppercase. To get proper folding of these characters and to allow them as part of an unquoted identifier, you can specify your own character set, which includes both classification and folding tables. Specify the CHARNAME parameter at startup to have the database manager use your character set as the default. You can then use characters such as o-umlaut or n-tilde in identifiers without the use of double quotation marks.

For information on how to define your own character set, see Appendix E, “Defining Your Own Character Set,” on page 363.

Hexadecimal Values of the Sample Character Sets

You will probably be able to use one of the IBM-supplied sample character sets without modification. This section shows the hexadecimal value that is used to represent each valid character. If your devices use those hexadecimal values for the indicated characters, you can use the IBM-supplied samples.

The ENGLISH character set is shown in Figure 95 on page 236. Only those characters that are identifiable by the database manager are shown. Any hexadecimal code that does not have a character assigned to it is unusable for DB2 Server for VSE keywords or unquoted identifiers. Such characters are usable in quoted identifiers and in constants and, of course, can be stored in the database.

For example, many display devices using an English character set assign a cent sign (¢) to X'4A'. In Figure 95 on page 236, however, no character is shown for the value X'4A' meaning that X'4A' is unusable for DB2 Server for VSE keywords or unquoted identifiers. If you want to put a cent sign in an identifier, you must use a delimited identifier.

Another example is the tilde (~). In most ENGLISH character sets, the tilde is represented by X'A1'. The matrix shows no entry for X'A1'. So, regardless of what X'A1' represents in your character set, you must use a delimited identifier.

These rules apply to the matrices for the other character sets as well. An important point to remember is that the absence of a character in one of the matrices does not prevent you from using that character set. The characters are not undefined to the database manager; they merely have limited use (as described above). Often, this limited use is exactly how you want the hexadecimal code to be handled. Independent of this qualification, you should almost always be able to find a CCSID that you can use at your installation. When you decide on a CCSID, try to avoid using a non-standard CCSID to prevent possible problems in the future (such as the inability to connect to other application servers because they do not support your CCSID). If you require a CCSID that is not supplied in the catalog tables, check the *Character Data Representation Architecture Level 1, Registry* manual for other predefined registered CCSIDs.

		00				01				10				11				Bits 0,1
	Hex 1	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
Bits 4567		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0					SP	&	–					^				0	
0001	1							/		a	j			A	J		1	
0010	2									b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5									e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9									i	r	z		I	R	Z	9	
1010	A						!		:									
1011	B					.	$	,	#									
1100	C					<	*	%	@									
1101	D					(	)	_	'									
1110	E					+	;	>	=									
1111	F					|		?	"									

Figure 95. ENGLISH Character Set (CCSID=37)

The sample FRENCH character set is shown in Figure 96 on page 237. Translation from lowercase to uppercase is done as follows:

X'6A' is translated to	X'E4'
X'7C'	X'C1'
X'C0'	X'C5'
X'D0'	X'C5'
X'E0'	X'C3'

These characters can be used in unquoted identifiers.

When evaluating the character set for use in your installation, remember that hexadecimal values that do not have characters assigned to them in Figure 96 on page 237 can be used in quoted identifiers.

Bits 0,1		00				01				10				11			
Bits 2,3		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
Bits 4567	Hex 1 / Hex 0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0					SP	&	–						é	è	ç	0
0001	1							/		a	j			A	J		1
0010	2									b	k	s		B	K	S	2
0011	3									c	l	t		C	L	T	3
0100	4									d	m	u		D	M	U	4
0101	5									e	n	v		E	N	V	5
0110	6									f	o	w		F	O	W	6
0111	7									g	p	x		G	P	X	7
1000	8									h	q	y		H	Q	Y	8
1001	9									i	r	z		I	R	Z	9
1010	A							ù	:				¬				
1011	B					.	$	,	£				|				
1100	C					<	*	%	à								
1101	D					(	)	_	’								
1110	E					+	;	>	=								
1111	F					!	︿	?	”								

Figure 96. FRENCH Character Set (CCSID=297)

The sample GERMAN character set is shown in Figure 97 on page 238. Translation from lowercase to uppercase is done as follows:

X'4A' is translated to	X'4A'
X'5A'	X'5A'
X'6A'	X'E0'
X'A1'	X'A1'
X'C0'	X'4A'
X'D0'	X'5A'
X'E0'	X'E0'

These characters can be used in unquoted identifiers.

When evaluating the sample character set for use in your installation, remember that hexadecimal values that do not have characters assigned to them in Figure 97 on page 238 can be used in quoted identifiers.

Some translation from lowercase to uppercase does not cause a change in the hexadecimal value. For more information, see “Step 3: Determine Translation Characters” on page 374.

		00				01				10				11				Bits 0,1
	Hex 1	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
Bits 4567		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0					SP	&	–						ä	ü	Ö	0	
0001	1							/		a	j	ß		A	J		1	
0010	2									b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5									e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9									i	r	z		I	R	Z	9	
1010	A					Ä	Ü	ö	:				¬					
1011	B					.	$	,	#				|					
1100	C					<	*	%	§									
1101	D					(	)	_	'									
1110	E					+	;	>	=									
1111	F					!	︿	?	"									

Figure 97. GERMAN Character Set (CCSID=273)

The sample ITALIAN character set is shown in Figure 98 on page 239. Translation from lowercase to uppercase is done as follows:

X'5A' is translated to	X'C5'
X'6A'	X'D6'
X'79'	X'E4'
X'A1'	X'C9'
X'C0'	X'C1'
X'D0'	X'C5'
X'E0'	X'C3'

These characters can be used in unquoted identifiers.

When evaluating the sample character set for use in your installation, remember that hexadecimal values that do not have characters assigned to them in Figure 98 on page 239 can be used in quoted identifiers.

		00				01				10				11				Bits 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
Bits 4567	Hex 1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0					SP	&	–						à	è	ç	0	
0001	1							/		a	j	ì		A	J		1	
0010	2									b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5									e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9								ù	i	r	z		I	R	Z	9	
1010	A						é	ò	:				¬					
1011	B					.	$	,	£				|					
1100	C					<	*	%	à									
1101	D					(	)	_	’									
1110	E					+	;	>	=									
1111	F					!	︿	?	”									

Figure 98. ITALIAN Character Set (CCSID=280)

The sample KATAKANA character set is shown in Figure 99 on page 240.

When evaluating the sample character set for use in your installation, remember that hexadecimal values that do not have characters assigned to them in Figure 99 on page 240 can be used in quoted identifiers.

		00				01				10				11				First Hex Char. Bits 0,1
	Hex 1	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
Bits 4567		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0					SP	&	-			ｿ		^				0	
0001	1						ｪ	/	i	ｱ	ﾀ			A	J		1	
0010	2						ｫ	a	j	ｲ	ﾁ	ﾍ		B	K	S	2	
0011	3						ｬ	b	k	ｳ	ﾂ	ﾎ	t	C	L	T	3	
0100	4						ｭ	c	l	ｴ	ﾃ	ﾏ	u	D	M	U	4	
0101	5						ｮ	d	m	ｵ	ﾄ	ﾐ	v	E	N	V	5	
0110	6					ｦ	ｯ	e	n	ｶ	ﾅ	ﾑ	w	F	O	W	6	
0111	7					ｧ		f	o	ｷ	ﾆ	ﾒ	x	G	P	X	7	
1000	8					ｨ	ｰ	g	p	ｸ	ﾇ	ﾓ	y	H	Q	Y	8	
1001	9					ｩ		h		ｹ	ﾈ	ﾔ	z	I	R	Z	9	
1010	A						!		:	ｺ	ﾉ	ﾕ	ﾚ					
1011	B					.	¥	,	#	q	r	s	ﾛ					
1100	C					<	*	%	@	ｻ		ﾖ	ﾜ					
1101	D					(	)	_	'	ｼ	ﾊ	ﾗ	ﾝ					
1110	E					+	;	>	=	ｽ	ﾋ	ﾘ	ﾞ					
1111	F					|		?	"	ｾ	ﾌ	ﾙ	ﾟ					

Figure 99. JAPANESE (Katakana) Character Set (CCSID=290, the SBCS Component of CCSID 5026)

The sample SPANISH character set is shown in Figure 100 on page 241. Translation from lowercase to uppercase is done as follows:

X'6A' is translated to X'7B'

These characters can be used in unquoted identifiers.

When evaluating the sample character set for use in your installation, remember that hexadecimal values that do not have characters assigned to them in Figure 100 on page 241 can be used in quoted identifiers.

		00				01				10				11				Bits 0,1
	Hex 1	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
Bits 4567		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0					SP	&	–									0	
0001	1							/		a	j			A	J		1	
0010	2									b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5									e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9									i	r	z		I	R	Z	9	
1010	A							ñ	:				^					
1011	B					.	Pts	,	Ñ				!					
1100	C					<	*	%	@									
1101	D					(	)	_	'									
1110	E					+	;	>	=									
1111	F					|		?	"									

Figure 100. SPANISH Character Set (CCSID=284)

Specifying an IBM-Supplied Character Set at Run Time

If the hexadecimal codes in one of the sample character sets matched those used by your devices, you can specify the character set at run time. To use a character set, specify the CHARNAME parameter when starting the application server. The CHARNAME parameter is valid in both single and multiple user mode. For information on how to specify the CHARNAME parameter, see "Setting the Application Server Default CHARNAME and CCSIDs" on page 249. Examples of IBM-supplied sample character sets are:

- ARABIC
- CYRILLIC
- DANISH-NORWEGIAN
- E-INTERNATIONAL
- ENGLISH
- ESTONIAN
- FINNISH-SWEDISH
- FRENCH
- GERMAN
- GREEK
- GREEK-423
- HEBREW
- ICELANDIC

- INTERNATIONAL
- ITALIAN
- JAPANESE-ENGLISH
- KATAKANA
- KOREAN
- LAO
- S-CHINESE
- SPANISH
- T-CHINESE
- THAI
- UK-ENGLISH
- UKRAINIAN
- VIETNAMESE
- 290
- 833
- 836
- 870
- 930
- 939
- 1027
- 1112
- 28709.

Figure 101 shows example job control to start the application server. The CHARNAME parameter indicates that the database manager is to use the FRENCH sample character set, and a CCSID of 297.

```
// JOB START SQL
// EXEC PROC=ARIS75PL
// EXEC PROC=DBNAME01
// EXEC PGM=ARISQLDS,SIZE=AUTO,PARM='PARMID=WARM1,CHARNAME=FRENCH'
/*
/&
```

Figure 101. Starting the Application Server to Use the French Character Set

The default character sets ENGLISH (CCSID=37) and INTERNATIONAL (CCSID=500) are hard coded into this product. For example, if you specify ENGLISH for the CHARNAME parameter, the database manager uses the ENGLISH character set that is coded internally. The internally coded character set is used even if a row exists in SYSTEM.SYSCHARSETS that has ENGLISH or INTERNATIONAL in its NAME column. (Neither the sample ENGLISH character set nor the sample INTERNATIONAL character set is used, although you can load either into SYSTEM.SYSCHARSETS. They are provided to make the definition of your own character sets easier.)

If you specify the name of a character set that is not defined in SYSTEM.SYSCHARSETS, the database manager displays an error message and uses the character set that was specified previously. If the character set is defined incorrectly in SYSTEM.SYSCHARSETS, an error message is displayed, and the database manager uses the character set that was previously specified.

Using Double-Byte Character Set (DBCS)

The double-byte character set (DBCS) option supports the use of DBCS characters in identifiers, constants and data. Identifiers can be either:
- Host identifiers (such as host variables), or
- SQL identifiers (such as dbspaces, tables or columns).

Constants and data containing DBCS characters can be either:
- Graphic data, or
- Character data with a mixed subtype (that is, character data containing DBCS characters).

Setting the DBCS option also ensures that:
- A shift-out character is paired with a shift-in character on output,
- When mixed character data is truncated, truncation does not occur between the two bytes of a DBCS character.

If your installation uses a double-byte character set, you should consider setting the DBCS option to YES.

For information on enabling the DBCS option, see "Setting the DBCS Option for the Application Server" on page 254, and "Setting the Default Application Requester DBCS Option" on page 254.

If you plan to use the double-byte character set (DBCS) characters, consider the following:
- PL/I programs using DBCS require no additional preprocessing (the PL/I compiler supports DBCS).
- DBCS variables and constants are not supported in FORTRAN and Assembler programs. However, you can use DBCS in dynamically defined SQL statements.
- SQL identifiers, SQL host variables, and SQL labels with DBCS characters can be used in SQL statements in COBOL II Release 2 or later. The COBOL Kanji Preprocessor is not required.

When the DBCS option is set to YES, the shift-out (X'0E') and shift-in (X'0F') delimiters are recognized in both identifiers in SQL statements and mixed data character string constants. The recognition of the delimiters provides the following benefits:
- On the application server
  - SQL identifiers can contain DBCS characters.
- On the application requester
  - Host identifiers can contain DBCS characters.
  - The DBS utility processing ensures the pairings of shift-out and shift-in characters.
  - ISQL allows the input, print, and display of DBCS characters and mixed data.

However, setting the DBCS option induces overhead for checking the proper pairing of shift-out and shift-in characters.

Identifiers Containing DBCS Characters

Identifiers can be either host identifiers or SQL identifiers.

To use host identifiers that contain DBCS characters, the VSE application requester must have the DBCS option set to YES. For more information, see “Setting the DBCS Option for the Application Server” on page 254 and “Setting the Default Application Requester DBCS Option” on page 254.

To use ordinary SQL identifiers that contain DBCS characters, the application server must have the DBCS option in the SYSTEM.SYSOPTIONS catalog table set to YES, and must also support DBCS characters and mixed data. The application server supports DBCS characters and mixed data when a mixed CHARNAME is specified as an initialization parameter. A mixed CHARNAME has a non-zero value for the CCSIDMIXED row in the SYSTEM.SYSOPTIONS catalog table. For more information, see “Choosing the Application Server Default CHARNAME and CCSID” on page 31.

If the DBCS option is set to YES for the application server, you can use DBCS characters in ordinary SQL identifiers. The identifier can be DBCS characters, or can contain a DBCS substring.

Identifiers are recorded in the catalog tables. When the database manager stores identifiers that contain DBCS characters, it also stores the shift-out and shift-in delimiters. The delimiters are stored because all columns of the catalog tables that contain identifiers have a data type of either CHAR or VARCHAR.

The number of bytes required to represent a string of DBCS characters is equal to:

```
2 x the number of DBCS characters + 2
```

For more information on how identifiers are used in application programs, see the *DB2 Server for VSE & VM Application Programming* manual.

Constants and Data Containing DBCS Characters

Constants and data containing DBCS characters can be either graphic data or character data with a mixed subtype.

To use graphic and mixed constants or data, the application server and the application requester must support mixed data. The application server supports graphic and mixed data when the default CHARNAME is a mixed CHARNAME. A mixed CHARNAME has a non-zero value for the CCSIDMIXED row in the SYSTEM.SYSOPTIONS catalog table. For more information, see “Choosing the Application Server Default CHARNAME and CCSID” on page 31. The VSE application requester supports graphic and mixed data when a mixed CHARNAME is set for the application requester. For more information, see “Setting the Application Requester Default CHARNAME and CCSIDs” on page 251. Using a mixed CHARNAME provides the following benefits:

- On the application server
  - Character string constants can contain mixed data consisting of DBCS and SBCS characters.
  - When character string constants containing DBCS characters are used in SQL statements, the data is correctly interpreted as having a mixed subtype and a mixed CCSID.
- On the application requester
  - Character string constants can contain mixed data consisting of DBCS, SBCS, and on some platforms, MBCS characters. (The MBCS characters will be stored as SBCS or DBCS characters by the database manager.)

The implied data type of all string character constants is VARCHAR. When the DBCS option is set to YES:

- Constants with only SBCS characters have a subtype of SBCS.
- Constants with a combination of SBCS and DBCS characters have a subtype of mixed. For example:

  ```
  'abc<XXYYZZ>'
  ```
- Constants containing only DBCS are also of the character data type with a subtype of mixed. They are *not* considered to be graphic constants. For example:

  ```
  '<XXYYZZ>'
  ```

When the DBCS option for the application server is set to NO, all character constants have a subtype of SBCS.

CCSID Conversion

Internally, character data is stored as hexadecimal values called code points. When a device interprets or displays a code point as a character, it uses a code page, which is a set of assignments of characters to code points. If two terminals use different code pages, they can display the same code point as a different character. For example, in code page 37, the code point X'4F' represents a vertical bar (|), but in code page 500, the code point X'4F' represents an exclamation mark (!).

As of Version 3 Release 4, the database manager supports Coded Character Set Identifiers (CCSIDs). The CCSID attribute specifies which code page to use both to map code points to characters, and to map characters to code points. The CCSID and the code points are used together to determine the character that the code point represents.

Note: The default CCSID is implicitly set by the default CHARNAME.

For example, suppose a DB2 Server for VSE online application requester has the default CHARNAME set to ENGLISH (CCSIDSBCS=37), the remote DRDA DB2 Server for VSE application server has the default CHARNAME set to INTERNATIONAL (CCSIDSBCS=500). In this case, if the user inserts an exclamation mark into a character column, the application requester sends X'5A' (the code point that represents an exclamation mark in the code page used with CCSID 37). The application server converts X'5A' to X'4F' (because X'4F' represents an exclamation mark in the code page used with CCSID 500), then stores X'4F' in the column.

If another application requester using a different CCSID retrieves the character, X'4F' is converted to the code point that represents an exclamation mark in the code page specified by the application requester CCSID. The character is interpreted and displayed correctly, and the hexadecimal value that is stored in the database is not changed.

For more information on how to decide the default CCSID values you should use, see “Determining CCSID Values” on page 248.

The sections that follow discuss the following topics:

- How to determine the CCSID values
- How to set the application server default CHARNAME and CCSID values
- How to set the application requester default CHARNAME and CCSID values
- How to set the application server default character subtype

- How to set the DBCS option for the application servers
- How to set the DBCS option for application requesters
- EUC Conversion.

For examples that show the interactions among the different values, see “Examples of Setting Values for an Installation” on page 256.

If an application requester and an application server do not use the same default CCSID, CCSID conversion is done during communications between the two.

Note: For an application requester using an ASCII representation of the data, CCSID conversion **always** occurs.

The application requester CCSIDs are recognized by the application server when DRDA support is installed and being used. If DRDA support is not installed, the application requester CCSIDs are not recognized by the application server.

Table 20 and Table 21 on page 247 show CHARNAMEs and the corresponding CCSIDs that can be used as system defaults. Table 20 shows the SBCS CHARNAME CCSIDs, and Table 21 on page 247 shows the mixed CHARNAME CCSIDs, with the component SBCS and DBCS CCSIDs for each mixed CCSID.

Table 20. SBCS CCSIDs

CCSID	Character Set	Code Page	CHARNAME	Description
37	697	37	ENGLISH	Country/region extended code pages (CECP): USA, Canada (S/370* system), Netherlands, Portugal, Brazil, Australia, New Zealand
273	697	273	GERMAN	CECP: Austria, Germany
277	697	277	DANISH-NORWEGIAN	CECP: Denmark, Norway
278	697	278	FINNISH-SWEDISH	CECP: Finland, Sweden
280	697	280	ITALIAN	CECP: Italy
284	697	284	SPANISH	CECP: Spain, Latin America (Spanish)
285	697	285	UK-ENGLISH	CECP: United Kingdom
290	1172	290	290	Japanese Katakana, extended host single byte
297	697	297	FRENCH	CECP: France
420	235	420	ARABIC	Arabic (all presentation shapes)
423	218	423	GREEK-423	Greek (Coexistence)
424	941	424	HEBREW	Hebrew
500	697	500	INTERNATIONAL	CECP: Belgium, Canada (AS/400* system), Switzerland, International Latin-1
833	1173	833	833	Korean, extended host single byte
836	1174	836	836	Simplified Chinese, extended host single byte
838	1176	838	THAI	Thai, extended host single byte
870	959	870	870	ROECE (Regional Office for East & Central Europe) Latin-2 Multilingual
871	697	871	ICELANDIC	CECP: Iceland

Table 20. SBCS CCSIDs (continued)

CCSID	Character Set	Code Page	CHARNAME	Description
875	925	875	GREEK	Greek
1025	1150	1025	CYRILLIC	Cyrillic Multilingual Turkish Latin 5
1027	1172	1027	1027	Japanese Latin, extended host single byte
1112	1305	1112	1112	Latvian/Lithuanian
1122	1307	1122	ESTONIAN	Estonian
1123	1326	1123	UKRAINIAN	Cyrillic Ukrainian EBCIDIC
1130	1336	1130	VIETNAMESE	EBCIDIC Vietnamese
1132	1341	1133	LAO	EBCIDIC Lao
1137	1137	1137	HINDI	Hindi
1142	697	1142	EDANISH-NORWEIGAN	Danish and Norwegian Euro CECP
1143	697	1143	EFINNISH-SWEDISH	Finnish and Swedish Euro CECP
1145	697	1145	E-SPANISH	Spanish Euro CECP
1148	697	500	E-INTERNATIONAL	International Euro CECP
1140	697	37	E-ENGLISH	English Euro CECP
1141	697	273	E-GERMAN	German Euro CECP
1144	697	280	E-ITALIAN	Italian Euro CECP
1146	697	285	E-UK-ENGLISH	UK English Euro CECP
1147	697	297	E-FRENCH	French Euro CECP
28709	1175	37	28709	Traditional Chinese, extended host single byte

Table 21. Mixed CCSIDs

Mixed	Component CCSIDs	Character Set	Code Page	CHARNAME	Description
930	290 (SBCS) 300 (DBCS)	1172 1001	290 300	930	Japanese (Katakana)-Kanji mixed host (including 4370 user-defined characters) extended single byte
933	833 (SBCS) 834 (DBCS)	1173 934	833 834	KOREAN	Korean host mixed (including 1880 user-defined characters) extended single byte
935	836 (SBCS) 837(DBCS)	1174 937	836 837	S-CHINESE	Simplified Chinese host mixed (1880 user-defined characters) extended single byte
937	28709 (SBCS) 835 (DBCS)	1175 935	37 835	T-CHINESE	Traditional Chinese host mixed (6204 user-defined characters) extended single byte
939	1027 (SBCS) 300 (DBCS)	1172 1001	1027 300	939	Japanese (Latin)-Kanji mixed host (including 4370 user-defined-characters) extended single byte
1364	833 (SBCS) 834 (DBCS)	65535 65535	833 834	KOREAN-1364	Korean host mixed extended including 11,172 full hangul

Table 21. Mixed CCSIDs (continued)

Mixed	Component CCSIDs	Character Set	Code Page	CHARNAME	Description
1388	836 (SBCS) 837 (DBCS)	65535 65535	846 837	S-CHINESE-GBK	S-Ch DBCS-Host Data GBK mixed, all GBK character set and other growing chars
5026	290 (SBCS) 4396 (DBCS)	1172 370	290 300	KATAKANA	Japanese (Katakana)-Kanji mixed host (including 1880 user-defined characters) extended single byte
5035	1027 (SBCS) 4396 (DBCS)	1172 370	1027 300	JAPANESE-ENGLISH	Japanese (Latin)-Kanji mixed host , (including 1880 user-defined characters) extended single byte

For more information about CCSIDs, see the *Character Data Representation Architecture Level 1, Registry*, and the *Character Data Representation Architecture Reference and Registry* manuals.

For information on the types of DBCS conversion that can be done between CCSIDs, see “Coding Your Own TRANSPROC Exit” on page 284.

Determining CCSID Values

When displaying characters on your terminal display, the default CCSID values for an application requester must be compatible with the code page that was used to generate its terminal controller. If the defaults are incompatible, characters will not be displayed or interpreted as expected, and the results of queries or inserts of either character or graphic data will be unreliable. Table 20 on page 246 and Table 21 on page 247 show the code pages that are compatible with each CCSID, and the CHARNAME value that you should specify. For example, if the controller was generated with code page 37, you should specify ENGLISH for the CHARNAME parameter. This value sets the value of CCSIDSBCS to 37, and the values of CCSIDMIXED and CCSIDGRAPHIC to 0. You can use the CICS DSQG transaction to set default values for all application requesters. For more information, see “Setting the Application Requester Default CHARNAME and CCSIDs” on page 251. The system-wide defaults may not be suitable for all application requesters: some may have a controller generated to use a code page incompatible with the system-wide default CHARNAME. In this situation, you should set the CHARNAME parameter for individual application requesters. For more information on this topic, see “Setting the Application Requester Default CHARNAME and CCSIDs” on page 251.

The default CCSID values for the application server can be set to any value you want, but keep in mind that you may want to set the default CCSIDs for the application server to values that can be used as defaults by the majority of the application requesters. This can reduce the amount of CCSID conversion that will be necessary. Also, consider setting the application server default CCSIDs to values that are compatible with the code page used to generate the terminal controller of the operator console. This ensures that data in single user mode applications is processed correctly (for example, single user mode DBS utility). Table 20 on page 246 and Table 21 on page 247 show the code pages compatible with each CCSID, and the CHARNAME that should be specified. For example, if the terminal controller of the operator console was generated with code page 37, you should use ENGLISH as the CHARNAME. This value sets the CCSIDSBCS value to 37, and the CCSIDMIXED and CCSIDGRAPHIC values to 0.

In particular, the CHARNAME INTERNATIONAL (CCSID=500) warrants special attention. This CHARNAME is composed of a code page that supports all the characters that are supported by the Latin-1 countries / regions, including Australia, Austria, Belgium, Brazil, Denmark, Canada, the Faroe Islands, Finland, France, Germany, Hong Kong, Iceland, Italy, Japan, the Netherlands, New Zealand, Norway, Portugal, Spain, Latin America, Sweden, Switzerland, the United Kingdom, and the United States. If all application requesters and application servers in these countries / regions use this CCSID, then single-byte CCSID conversion will not be necessary for accessing data from different sites. This may provide savings because of lower CPU usage.

Often, it may not be appropriate to use CCSID 500 at every site. For example, you may have to use existing equipment (such as terminal controllers that use other character sets and code pages), or you may have a large quantity of data that is stored using a CCSID other than 500. However, if you plan to frequently access data from other countries / regions, you should consider migrating your data and hardware to CCSID 500, both for performance reasons, and for the ease of data management.

Setting the Application Server Default CHARNAME and CCSIDs

The different uses of the default system CCSIDs are shown in "Choosing the Application Server Default CHARNAME and CCSID" on page 31. Data in columns which were migrated from a release earlier than Version 3 Release 4 have a CCSID which is obtained from rows in the SYSTEM.SYSOPTIONS catalog table. These rows are: MCCSIDSBCS, MCCSIDMIXED and MCCSIDGRAPHIC. For more information on the SYSTEM.SYSOPTIONS catalog table, see the *DB2 Server for VSE & VM SQL Reference* manual. To change either the application server default CCSIDs or the CCSIDs that are used for data in migrated columns, you must have DBA authority.

The only way to change the system CCSID is to change the CHARNAME parameter when you start the application server. This also updates the following columns of the SYSTEM.SYSOPTIONS catalog table: CCSIDSBCS, CCSIDMIXED and CCSIDGRAPHIC. For more information, see "Character Set Considerations at Startup" on page 51.

You may have to use different default CCSIDs for columns that were created before the migration than for columns created after the migration. For example, suppose that you are migrating your database and want to use the INTERNATIONAL character set (CCSID=500) for character columns that were created after the migration. Character columns that existed before migration were created with the ENGLISH character set (CCSID=37). To ensure that the data in existing character columns is displayed and interpreted correctly, (that is, as was done before the migration), you require an MCCSIDSBCS value of 37, and a CCSIDSBCS value of 500.

Be very careful when using different default CCSIDs. This should only be done when there is a specific reason for them to be different. When the application server and application requester have different CCSID values, character conversion cannot be avoided. This conversion has an associated performance overhead, and causes performance degradation. CCSID conversion of data also affects the sargability of predicates. For more information on performance, see the *DB2 Server for VSE & VM Performance Tuning Handbook*.

Note: Use caution when you change the application server default CCSIDs. For more information, see "Determining CCSID Values" on page 248.

For many characters, the corresponding hexadecimal value in the International code page is the same as in the English code page. However, this is not true of all characters. For example, in the English code page the hexadecimal value corresponding to the exclamation mark (!) is '5A', but in the International code page the value is '4F'. Table 22 lists the differences between the International code page and the English code page.

Table 22. Differences between International Code Page and English Code Page

Character	CCSID=37	CCSID=500
^	X'B0'	X'5F'
¢	X'4A'	X'B0'
!	X'5A'	X'4F'
[	X'BA'	X'4A'
]	X'BB'	X'5A'
|	X'4F'	X'BB'
¬	X'5F'	X'BA'

For more information on code page details, see the *Character Data Representation Architecture Level 1, Registry* manual.

Columns must be tagged with the CCSID that corresponds to the code page with which they were created or the results of queries on these columns will be unreliable. For example, suppose that the following column was created with the English character set before migration:

```
CHARDATA
--------
ABCDEFGH
    kjp!
    ¬ds!
```

If MCCSIDSBCS is 37 (corresponding to English), and CCSIDSBCS is 500 (corresponding to International), performing a SELECT operation on this column after the migration gives the results shown above. However, if MCCSIDSBCS is incorrectly set to 500 (corresponding to the International character set), performing a SELECT operation on the column produces the following result:

```
CHARDATA
--------
ABCDEFGH
    kjp]
    ^ds]
```

In this example, to ensure reliable results, MCCSIDSBCS must be 37, regardless of the value of CCSIDSBCS.

Changing the CCSID Attribute of an Existing Column

If you want to change the CCSID attribute of an existing column, use the DBS utility. For example, to change the default CCSID for data in columns that were created previous to the migration to Version 3 Release 4, use the DBS utility to do the following:

1. Unload the data from the existing table.
2. Drop the table.
3. Recreate the table, specifying the new CCSID attribute for the column or columns that you want to change, or use the default if it is appropriate.
4. Reload the data.

Note: You must use the DBSU ″DATALOAD/DATAUNLOAD″ commands, **NOT** the ″UNLOAD/RELOAD″ commands.

Changing the Subtype Attribute of an Existing Column

The subtype attribute is only used when the CCSID attribute is null. If you have migrated from a release previous to Version 3 Release 4, existing character columns will have a CCSID value of null. For these columns, the subtype value is used to indicate their CCSID value. The CCSID value is either the value for MCCSIDSBCS (for a subtype of “S”) or the value for MCCSIDMIXED (for a subtype of “M”).

In some cases, columns with a null CCSID could have a subtype of “S” and contain mixed data. This can occur if the column was created without specifying the FOR MIXED DATA clause. In this case, the subtype must be changed to “M” in order for the correct CCSID to be used for this column. Otherwise, conversion errors can occur (for example, SQLCODE -330, SQLSTATE 22517).

To change the subtype, DBA authority is required to update the SYSTEM.SYSCOLUMNS table. Change the value in the SUBTYPE column from “S” to “M” for the required character column.

Setting the Application Requester Default CHARNAME and CCSIDs

VSE application requesters use the CHARNAME that is specified in the SQLGLOB file, either a specific userid’s value or the global default value as follows:

- Batch applications use the SQLGLOB file ’CHARNAME’ setting associated with the userid(s) contained on the SQL CONNECT statement(s) used in the application. If any of these userids are NOT defined in the SQLGLOB file, then the global CHARNAME will be used.
- The preprocessor is a batch application requester. The userid given by the ’USERID’ preprocessor parameter will be used to determine the CHARNAME from the SQLGLOB file. If the userid is not defined in the SQLGLOB file, the global default CHARNAME value will be used.
- The Database Services Utility is a batch application requester. It uses the SQLGLOB file global CHARNAME setting to determine the input data folding translation table to be used. Otherwise, it will use the SQLGLOB file ’CHARNAME’ setting associated with the userid(s) contained on the SQL CONNECT statement(s) used in the application. If any of these are NOT defined in the SQLGLOB file, then the global CHARNAME will be used.
- ISQL is an online application requester. It has a CICS user ID associated with it. Therefore, ISQL uses the user setting of the SQLGLOB parameter CHARNAME

if the ISQL user had used the DSQU transaction to override the global setting of the CHARNAME parameter. Otherwise, ISQL uses the global setting of the SQLGLOB parameter CHARNAME.

Note: If ISQL is running against a remote DRDA application server, the ISQL user must ensure that the global and the user setting of the SQLGLOB parameter CHARNAME are the same. ISQL has a two transaction structure: ISQL and CISQ. ISQL starts CISQ. The former controls the terminal and the latter is for access to the application server.

Because CISQ is a transaction started by ISQL, it does not have a CICS user ID associated with it. Therefore, the online DRDA resource adapter uses the global setting of the SQLGLOB parameter CHARNAME to process the SQL requests from CISQ. On the other hand, ISQL, being the front-end transaction, has a CICS user ID associated with it. Therefore, ISQL uses the user setting of the SQLGLOB parameter CHARNAME, if available, to process the same SQL statement it received from the terminal.

- All ″started″ CICS transactions that pass SQL requests to a remote DRDA application server do not have a CICS user ID associated with it. Therefore, these transactions will use the global setting of the SQLGLOB parameter CHARNAME.
- All online DRDA application requesters have a CICS user ID associated with it. Therefore, they will use the user setting of the SQLGLOB parameter CHARNAME if the CICS user running the online DRDA transaction had used the DSQU transaction to override the global setting of the CHARNAME parameter. Otherwise, they will use the global setting of the SQLGLOB parameter CHARNAME.

If you want to check the global setting for the SQLGLOB parameter CHARNAME, use the DSQQ transaction without specifying any parameter. The CHARNAME displayed is the global setting of the SQLGLOB parameter CHARNAME.

If you want to check the CHARNAME setting for a CICS user, use the DSQQ transaction and specify the user ID of the CICS user. If the user setting exists, it will be displayed. If the user setting does not exist, a message will be displayed.

For more information about the DSQQ transaction, see the *DB2 Server for VSE & VM Database Administration*.

If you want to use the IBM-supplied default global settings for the SQLGLOB parameters (including the default for CHARNAME), you must execute the IBM-supplied job control program ARISGDEF. This program will initialize the global settings for the SQLGLOB parameters with IBM-supplied default values. (For more information on the default values for the SQLGLOB parameters, see the *DB2 Server for VSE & VM Database Administration* manual.)

If you want to use your own default global settings for the SQLGLOB parameters (including the default for CHARNAME), you must first execute the IBM-supplied job control program ARISGDEF and then execute the DSQG transaction.

To specify ENGLISH as the new global setting for the SQLGLOB parameter CHARNAME, run the DSQG transaction and specify the ENGLISH CHARNAME value as follows:

```
DSQG ENGLISH
```

The global settings for the SQLGLOB parameters do not apply to online application requesters that have already run the DSQU transaction (and thereby have their own user SQLGLOB parameter values).

For more information about the DSQG transaction, see the *DB2 Server for VSE & VM Database Administration*.

To specify a user setting for the SQLGLOB parameter CHARNAME which is different from the default global setting, run the DSQU transaction and specify the CHARNAME value. This value overrides the global CHARNAME.

For example, to specify ENGLISH as the new user setting for the SQLGLOB parameter CHARNAME for the CICS user CICSUSER, CICSUSER must sign on to a CICS session, run the DSQU transaction and specify the ENGLISH CHARNAME value as follows:

```
DSQU ENGLISH
```

For more information about the DSQU transaction, see the *DB2 Server for VSE & VM Database Administration*.

The SQLGLOB File Batch Query/Update Program

This program can be used to query and update the SQLGLOB file in batch mode, using the JCL in library member ARISBGUD.Z. It may be necessary to use this program to add or change SQLGLOB parameters when CICS and the DSQx transactions are not available, or for userids that do not exist under CICS, but are in batch jobs that access remote servers. With this program, any userid's SQLGLOB parameters can be queried, inserted, updated or deleted. The Global Default parameters cannot be deleted. For more information about this batch program, see the *DB2 Server for VSE & VM Database Administration*.

Setting the Application Server Default Character Subtype

To set the application server default character subtype, you must update a row in the SYSTEM.SYSOPTIONS catalog table. You must have DBA authority to do so. The CHARSUB option specifies the default subtype for a column when SUBTYPE clause or the CCSID is not specified explicitly (for example, on a CREATE TABLE or ALTER TABLE statement).

Note: The character subtype is defined for the application server only. It is not defined for the application requester. The CREATE PACKAGE CHARSUB option or the preprocessor CHARSUB option defines the default subtype for a package. For more information on this option, see the *DB2 Server for VSE & VM Application Programming* or the *DB2 Server for VSE & VM SQL Reference* manuals.

The initial setting of the application server default character subtype is SBCS. To set it to mixed, issue:

```
UPDATE SYSTEM.SYSOPTIONS
  SET VALUE = 'MIXED'
  WHERE SQLOPTION = 'CHARSUB'
```

To reset the application server default character subtype to SBCS, issue:

```
UPDATE SYSTEM.SYSOPTIONS
  SET VALUE = 'SBCS'
  WHERE SQLOPTION = 'CHARSUB'
```

In both situations, the new setting does not become effective immediately. The new setting is not in effect until the next time the application server is started.

If anything other than 'MIXED' or 'SBCS' is specified for the application server default character subtype in the SYSOPTIONS table, SBCS is assumed and an error message is issued when the application server is started.

The application server default character subtype can only be mixed when the application server default CHARNAME is mixed. The application server default character subtype is forced to be SBCS when the application server default CHARNAME is an SBCS CHARNAME.

Setting the DBCS Option for the Application Server

The DBCS option is set by updating a particular row in the SYSTEM.SYSOPTIONS catalog table. The initial setting of the DBCS option is NO. To set the DBCS option to YES, issue:

```
UPDATE SYSTEM.SYSOPTIONS
  SET VALUE = 'YES'
  WHERE SQLOPTION = 'DBCS'
```

To reset the DBCS option to NO, issue:

```
UPDATE SYSTEM.SYSOPTIONS
  SET VALUE = 'NO'
  WHERE SQLOPTION = 'DBCS'
```

You must have DBA authority to issue either of the above commands.

The new setting of the DBCS option will be effective the **next** time that the application server is started. The new setting does not become effective immediately.

If anything other than YES or NO is specified for the DBCS option in the SYSOPTIONS table, NO is assumed, and an error message is issued during startup.

For more information, see “Using Double-Byte Character Set (DBCS)” on page 243.

Setting the Default Application Requester DBCS Option

The VSE application requester DBCS option is set by the DBCS option contained in the SQLGLOB file, as follows:

- Batch applications use the SQLGLOB file ’DBCS’ option associated with the userid(s) contained on the SQL CONNECT statement(s) used in the application. If any of these userids are NOT defined in the SQLGLOB file, then the global DBCS will be used.
- The preprocessor is a batch application requester. The userid given by the ’USERID’ preprocessor parameter will be used to determine the DBCS option from the SQLGLOB file. If the userid is not defined in the SQLGLOB file, the global default DBCS value will be used.
- The Database Services Utility is a batch application requester. It uses the SQLGLOB file global DBCS value.
- ISQL is an online application requester. It has a CICS user ID associated with it. Therefore, ISQL uses the user setting of the SQLGLOB parameter DBCS if the

ISQL user had used the DSQU transaction to override the global setting of the DBCS parameter. Otherwise, ISQL uses the global setting of the SQLGLOB parameter DBCS.

Note: If ISQL is running against a remote DRDA application server, the ISQL user must ensure that the global and the user setting of the SQLGLOB parameter DBCS are the same. ISQL has a two transaction structure: ISQL and CISQ. ISQL starts CISQ. The former controls the terminal and the latter is for access to the application server.

Because CISQ is a transaction started by ISQL, it does not have a CICS user ID associated with it. Therefore, the online DRDA resource adapter uses the global setting of the SQLGLOB parameter DBCS to process the SQL requests from CISQ. On the other hand, ISQL, being the front-end transaction, has a CICS user ID associated with it. Therefore, ISQL uses the user setting of the SQLGLOB parameter DBCS, if available, to process the same SQL statement it received from the terminal.

- All ″started″ CICS transactions that pass SQL requests to a remote DRDA application server do not have a CICS user ID associated with it. Therefore, these transactions will use the global setting of the SQLGLOB parameter DBCS.
- All online DRDA application requesters have a CICS user ID associated with it. Therefore, they will use the user setting of the SQLGLOB parameter DBCS if the CICS user running the online DRDA transaction had used the DSQU transaction to override the global setting of the DBCS parameter. Otherwise, they will use the global setting of the SQLGLOB parameter DBCS.

If you want to check the global setting for the SQLGLOB parameter DBCS, use the DSQQ transaction without specifying any parameter. The DBCS displayed is the global setting of the SQLGLOB parameter DBCS.

If you want to check the DBCS setting for a CICS user, use the DSQQ transaction and specify the user ID of the CICS user. If the user setting exists, it will be displayed. If the user setting does not exist, a message will be displayed.

For more information about the DSQQ transaction, see the *DB2 Server for VSE & VM Database Administration*.

If you want to use the IBM-supplied default global settings for the SQLGLOB parameters (including the default for DBCS), you must execute the IBM-supplied job control program ARISGDEF. This program will initialize the global settings for the SQLGLOB parameters. (For more information on the default values for the SQLGLOB parameters, see the *DB2 Server for VSE & VM Database Administration* manual.)

If you want to use your own default global settings for the SQLGLOB parameters (including the default for DBCS), you must first execute the IBM-supplied job control program ARISGDEF and then execute the DSQG transaction.

To specify YES as the new global setting for the SQLGLOB parameter DBCS, run the DSQG transaction and specify the YES DBCS value as follows:

```
DSQG ,,YES
```

The global settings for the SQLGLOB parameters do not apply to online application requesters that have already run the DSQU transaction (and thereby have their own user SQLGLOB parameter values).

For more information about the DSQG transaction, see the *DB2 Server for VSE & VM Database Administration* manual.

To specify a user setting for the SQLGLOB parameter DBCS which is different from the default setting, run the DSQU transaction and specify the DBCS value. This value overrides the global setting for the user who ran the DSQU transaction.

For example, to specify YES as the new user setting for the SQLGLOB parameter DBCS for the CICS user CICSUSER, CICSUSER must sign on to a CICS session, run the DSQU transaction and specify the YES DBCS value as follows:

```
DSQU ,,YES
```

For more information about the DSQU transaction, see the *DB2 Server for VSE & VM Database Administration* manual.

The SQLGLOB File Batch Query/Update Program can also be used to display or modify user or global default parameters. For more information, see “The SQLGLOB File Batch Query/Update Program” on page 253, or the *DB2 Server for VSE & VM Database Administration*.

EUC Conversions

Extended UNIX™ Code (EUC) allows for a form of ASCII mixed data. It is an encoding scheme supported by UNIX in far eastern countries / regions which allows for MBCS characters. Each EUC codepage is made up of three character sets, or planes, denoted by G0, G1, and G2 or four character sets, denoted by G0, G1, G2 and G3. The group in which the data belongs is determined by the range of its first and second bytes. G0 is comprised of single-byte characters and is the ASCII invariant coded character set. G1 characters are double-byte characters within another range. G2 and G3 characters are triple-byte characters, distinguished by the first byte and the range of the last three bytes.

EUC conversion is supported by the database manager. EUC characters are converted to SBCS or DBCS characters, or both.

Unicode Conversions

Unicode data cannot be stored in a DB2 Server for VSE & VM database. However, Unicode data can be received, converted to certain host code pages, and then stored. To determine which Unicode conversions are supported, refer to the system catalog SYSTEM.SYSSTRINGS. Each row in SYSTEM.SYSSTRINGS represents a supported CCSID conversion, where INCCSID is the input CCSID and OUTCCSID is the CCSID to which it can be converted.

Examples of Setting Values for an Installation

This section discusses two examples of using the application server default CHARNAME JAPANESE-ENGLISH (CCSID=5035). The first example shows how to specify this CHARNAME and enable mixed string manipulation. The second example shows how to specify this CHARNAME without enabling mixed string manipulation and how to prevent the verification of character strings that contain mixed data. (Mixed string manipulation is the ability to specify mixed SQL identifiers, such as columns.)

Example 1

Suppose that you want to use the mixed JAPANESE-ENGLISH CCSID, 5035, as your application server default CCSID, and you also want to have the ability to do mixed string manipulation. To do this, set up your environment as follows:

1. Ensure that your terminal controllers are generated to use the correct code pages.

   The CCSID you want to use is 5035. You must define the controller to use the character set 1172 for the SBCS character set, and code page 1027 for the SBCS code page. For the DBCS characters, specify the character set 370 and the code page 300.
2. Install the database manager.

   The application server default CCSID for a newly installed database manager is 500 (CHARNAME=INTERNATIONAL). After installation, the SYSTEM.SYSOPTIONS catalog table contains the following information:
   ```
   CHARNAME=INTERNATIONAL (the name of 500)
   CCSIDSBCS=500
   CCSIDMIXED=0
   CCSIDGRAPHIC=0
   DBCS=NO
   CHARSUB=SBCS
   .............
   ```
3. Change the value of the application server default CHARNAME to JAPANESE-ENGLISH

   Start the application server. Specify CHARNAME=JAPANESE-ENGLISH. Message ARI0159D is displayed that informs you that the new CHARNAME (JAPANESE-ENGLISH) is different from the current default (INTERNATIONAL). You are prompted to enter either 1 (YES) to change the default, `0` (NO) to leave the default unchanged, or 111 (QUIT) to shut down the application server. Type 1 (for YES) and press ENTER.

   After the application server is started, the SYSTEM.SYSOPTIONS catalog table should contain the following information:
   ```
   CHARNAME=JAPANESE-ENGLISH
   CCSIDSBCS=1027      (the single-byte portion of 5035)
   CCSIDMIXED=5035
   CCSIDGRAPHIC=4396   (the double-byte portion of 5035)
   DBCS=NO
   CHARSUB=SBCS
   ................
   ```
4. To enable mixed string manipulation, change the value for DBCS in SYSTEM.SYSOPTIONS from NO to YES. You can use either ISQL or the DBS utility.
5. Because most of the character columns will contain mixed data, you should also change the value for CHARSUB from SBCS to MIXED.
6. To cause the DBCS and CHARSUB values in SYSTEM.SYSOPTIONS to be used as the new application server defaults, you must stop the application server, and then restart it.

   The changes are now complete. The SYSTEM.SYSOPTIONS catalog table contains the following information:
   ```
   CHARNAME=JAPANESE-ENGLISH
   CCSIDSBCS=1027      (the single-byte portion of 5035)
   CCSIDMIXED=5035
   CCSIDGRAPHIC=4396   (the double-byte portion of 5035)
   DBCS=YES
   CHARSUB=MIXED
   ................
   ```

7. To set these values for the online DRDA requesters, ISQL and the preprocessors, run the DSQG transaction. Issue the following command:
```
DSQG JAPANESE-ENGLISH,,YES
```

Example 2

Suppose that you want to use the mixed JAPANESE-ENGLISH CCSID, 5035, as your application server default CCSID. Because you must be able to both store DBCS characters, and retrieve DBCS characters from graphic columns (GRAPHIC, VARGRAPHIC, or LONG VARGRAPHIC), you cannot specify an ENGLISH single-byte CCSID such as 37 or 1027. Also suppose that you do not want the ability to do mixed string manipulation, and you want to prevent the database manager from verifying character strings for mixed data. In addition, you also want character columns that are created without the explicit specification of a CCSID or a subtype to default to the SBCS subtype and CCSID. To do this, set up your environment as follows:

1. Ensure that your terminal controllers are generated to use the correct code pages.

   The CCSID you want to use is 5035. You must define the controller to use the character set 1172 for the SBCS character set, and code page 1027 for the SBCS code page. For the DBCS characters, specify the character set 370 and the code page 300.
2. Install the database manager.

   The application server default CCSID for a newly installed database manager is 500 (CHARNAME=INTERNATIONAL). After installation, the SYSTEM.SYSOPTIONS catalog table contains the following information:
   ```
   CHARNAME=INTERNATIONAL (the name of 500)
   CCSIDSBCS=500
   CCSIDMIXED=0
   CCSIDGRAPHIC=0
   DBCS=NO
   CHARSUB=SBCS
   .............
   ```
3. Change the value of the application server default CHARNAME to JAPANESE-ENGLISH.

   Start the application server. Specify CHARNAME=JAPANESE-ENGLISH. Message ARI0159D is displayed that informs you that the new CHARNAME (JAPANESE-ENGLISH) is different from the current default (INTERNATIONAL). You are prompted to enter either `1` (YES) to change the default, `0` (NO) to leave the default unchanged, or `111` (QUIT) to shut down the application server. Type `1` (for YES) and press ENTER.

   After the application server is started, the SYSTEM.SYSOPTIONS catalog table should contain the following information:
   ```
   CHARNAME=JAPANESE-ENGLISH
   CCSIDSBCS=1027      (the single-byte portion of 5035)
   CCSIDMIXED=5035
   CCSIDGRAPHIC=4396   (the double-byte portion of 5035)
   DBCS=NO
   CHARSUB=SBCS
   .................
   ```
4. Because you do not want to enable mixed string manipulation, and you do not want the database manager to verify character strings for mixed data, leave the DBCS option set to NO (even though the database manager uses a mixed CCSID). This still allows you to:

- Issue CREATE TABLE or ALTER TABLE statements to either add or create GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC columns. The CCSID for these columns will be taken from value for CCSIDGRAPHIC in the SYSTEM.SYSOPTIONS catalog table.
- Insert into graphic columns from graphic host variables.
- Select graphic columns into graphic host variables.
- Use graphic constants in SQL statements.

5. Because most character columns will contain SBCS data, leave the value for CHARSUB as SBCS. When you need to either create or add a mixed character column, you can specify the FOR MIXED DATA clause or the CCSID clause explicitly for the CREATE TABLE or the ALTER TABLE statement.
6. To set these values for all the online DRDA application requesters, ISQL and the preprocessors, run the DSQG transaction. Issue the following command:

   ```
   DSQG JAPANESE-ENGLISH,,YES
   ```

   For an application requester to be able to use graphic data, the application requester must use a mixed CCSID as the default. One exception exists. If the application requester is connecting to a local application server, the application server responds to the application requester with the expectation that the application requester is using the same mixed CCSID as the application server is using. If the user specified a different value for the SQLGLOB CHARNAME parameter, the application server ignores this value. However, folding performed by the application requester is always based on the application requester's CHARNAME setting. For more information on the application requester's CHARNAME setting, see "Setting the Application Requester Default CHARNAME and CCSIDs" on page 251. In this case, if the application requester CHARNAME and the application server CHARNAME are not the same, unexpected results can occur.

Identifying Classification and Translation Tables for a CCSID

To identify either the classification table or the translation table that is used for folding characters to uppercase for a specific CCSID, do the following:

1. Review the CHARNAME column of the SYSTEM.SYSCCSIDS catalog table for the CHARNAME value of the CCSID.
2. Review the NAME column of the SYSTEM.SYSCHARSETS catalog table for the value that matches the CHARNAME of the CCSID. That row contains both the classification table and the translation table for the CCSID.

National Language Support for Messages and HELP Text

The database manager can provide DB2 Server for VSE messages and HELP text in several national languages. Messages and HELP text come with the product tape. For more information on HELP text, see the *DB2 Server for VSE & VM Database Administration* manual. Installation instructions for HELP text is in the *DB2 Server for VSE Program Directory* manual.

When the national language feature tape has been installed, national language support works this way:

- Users of ISQL, the DBS utility, and the preprocessors can receive messages in the language they select.
- ISQL users can receive HELP text for commands and messages in the language they select.

- The DB2 Server for VSE operator can receive messages on the operator console in the language selected. The VSE operator cannot choose a double-byte character set (DBCS) language, as the VSE operator console does not support DBCS.

The national language tape provided contains the following languages:
- Mixed American English
- Uppercase American English
- French
- German
- Japanese
- Simplified Chinese.

When the database manager is installed, you specify a default national language. This is a second-level default. For online users, the first-level default is the LANGID supplied on the CIRB transaction. The second-level default is used for batch/ICCF users, and online users when LANGID is not specified. When one or more additional national languages have been installed, users can change the language from the default in the following ways:
- ISQL users can choose the language for messages and HELP text using the SET LANGUAGE command. For ISQL users to receive HELP text in the language they choose, the messages and the HELP text for that language must be installed. To support a national language, you must install the messages for that language. Installing the HELP text is optional.

  The VSE online users can also choose the language they receive messages in by specifying the LANGID parameter on the CIRB transaction. For an explanation of the CIRB transaction, see “Starting the Online Resource Adapter -- The CIRB Transaction” on page 83.
- DBS utility users and preprocessor users receive messages in the language specified when the database manager was installed. If a user wants to receive messages in another language, the user should specify the library containing the desired language in the LIBDEF statement of the job control.
- The operator can choose the language for operator messages using the SET LANGUAGE command from the operator console. This language is also used to display the output of the SHOW, RESET, and COUNTER commands. The VSE operator cannot choose a double-byte character set (DBCS) language, as the VSE operator console does not support DBCS.

National languages are identified to the database manager by a language name, and a LANGID (language identifier). These values are in the SQLDBA.SYSLANGUAGE table. If you have English and French installed on the database manager, the SQLDBA.SYSLANGUAGE table can look like the example in Figure 102.

```
LANGUAGE        LANGKEY  REMARKS                                   LANGID
--------------  -------  ----------------------------------------  --------
ENGLISH            S001  ENGLISH VERSION OF HELP TEXT               AMENG
FRANCAIS           S003  TEXTE D'AIDE FRANCAIS                      FRANC
```

Figure 102. Sample SQLDBA.SYSLANGUAGE Table

For the LANGUAGE and REMARKS columns, you can choose values appropriate for your organization. For the LANGKEY and LANGID columns, you should keep the values supplied by the database manager.

The language keys (LANGKEY) and language identifiers (LANGID) used by the database manager are shown in Table 23.

Table 23. Language Keys and Language Identifiers

LANGUAGE	LANGKEY	LANGID
ENGLISH (mixed case)	S001	AMENG
ENGLISH (uppercase)	S002	UCENG
FRENCH	S003	FRANC
GERMAN	S004	GER
JAPANESE	D001	KANJI
CHINESE_SIMPLIFIED	D003	HANZI

You should not use the language keys and language identifiers (LANGID) shown above for other purposes. In addition, the language keys S007-S500 and D003-D500 are reserved for IBM use.

The language key is used to internally identify HELP text for a language. The LANGID can be used to choose a language for messages and HELP text. You can also specify the name of the language, as it is stored in the LANGUAGE column of the SQLDBA.SYSLANGUAGE table.

In ISQL, and on the operator console, you can specify a language or a LANGID on the SET LANGUAGE command. The VSE operator cannot choose a double-byte character set (DBCS) language, as the VSE operator console does not support DBCS. The syntax of the SET LANGUAGE command is shown in Figure 103.

```
►►──SET LANGuage──┬─language─┬──────────────────────────────────────────►◄
                  └─langid───┘
```

Figure 103. The SET LANGUAGE Command

The language or LANGID you specify must match a value in the SYSLANGUAGE table, and must be installed. If your installation uses a double-byte character set, you should consider setting the DBCS option to YES. For information on the DBCS option, see "Using Double-Byte Character Set (DBCS)" on page 243.

When using the LANGID parameter on the CIRB transaction, you can specify only the LANGID. At startup, messages are displayed in the default national language.

Changing the ISQL Default Language

The default language for ISQL in VSE is set using the LANGID parameter of the CIRB transaction. If the LANGID parameter is not specified, the default language for ISQL is the one specified in an internal table structure called LANGBLK. There is one LANGBLK table structure for each national language. The database manager uses the language specified in the ARIMLBK TEXT member. A name is assigned to each LANGBLK table structure for each language, as follows:

Table 24. LANGBLK Table

Mixed case English	ARIMLBKD

Table 24. LANGBLK Table (continued)

Uppercase English	ARIMLBKU
French	ARIMLBKF
German	ARIMLBKG
Japanese	ARIMLBKJ
Simplified Chinese	ARIMLBKC

To change the ISQL default language, rename the member for the language you want to use to ARIMLBKD, and relink-edit the ARISLKYD link book.

National Language Messages in a VSE Guest Sharing Environment

If you have VSE guest sharing, you should install all languages on the VSE guest that you want to support on VSE. Users who use the DBS utility and the preprocessors from the VSE guest should specify the library containing the desired language in the LIBDEF statement of the job control.

Chapter 13. Creating Installation Exits

This chapter discusses installation exits that:
- Supply account numbers for product users
- Define your own datetime format
- Coding your own TRANSPROC exit
- Perform your own cancel exit
- Encode and decode data (Field Procedures).

Supplying Account Numbers for Users

There is no rigid format for entering account or project numbers into accounting records, because their definition and use vary at each installation. (Some installations do not use account numbers at all.) Thus, you must devise your own scheme.

To do this, you replace a module named ARIUXIT with your own version of that module.

The resource adapter calls ARIUXIT when a user tries to connect to a DB2 Server for VSE application server either implicitly or explicitly. The database manager branches to ARIUXIT even before attempting to verify the user.

The database manager allows ARIUXIT to access a control block. In this control block, ARIUXIT can provide up to 16 bytes of data.

Before calling ARIUXIT, the database manager initializes the 16-byte area. For batch/ICCF applications, the database manager initializes the area to character blanks.

The ARIUXIT module does not use the control block (except for the return code area); it only sets a no-operation return code and branches back to the database manager, as shown in the following figure.

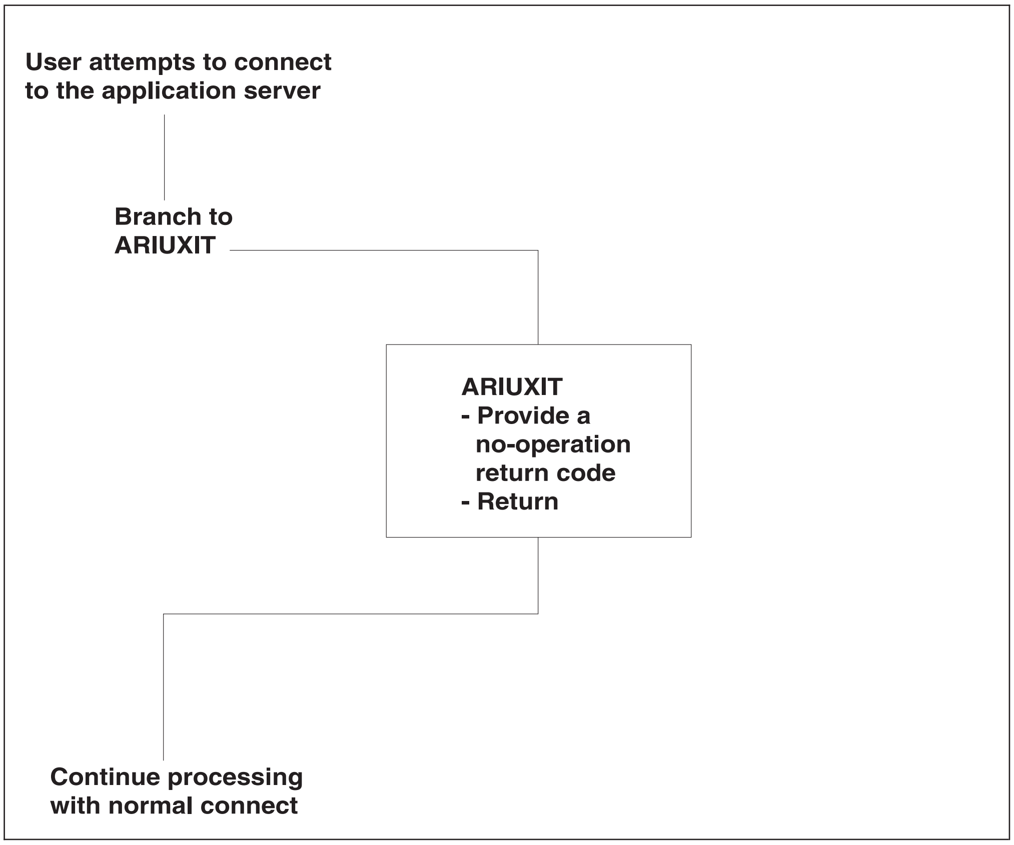

User attempts to connect to the application server
Branch to ARIUXIT
ARIUXIT
- Provide a no-operation return code
- Return
Continue processing with normal connect

Figure 104. The Database Manager Branching to ARIUXIT

Because of this, the database manager places blanks (for batch/ICCF) or CICS information (for online applications) in the installation-dependent field of the user’s accounting records. If you choose, you can change (write over) this area. Whatever data is in the 16-byte area is placed on the accounting records of the user who was trying to connect at the time that ARIUXIT was called.

Your version of ARIUXIT should determine the user’s accounting information for your installation, verify it, and pass it to database manager which puts it in the user’s accounting records. You can supply department names as well as account or project numbers. You can, in fact, supply whatever you like so long as it fits in 16 bytes and meets your own installation’s requirements. The database manager does no error-checking on the data.

The database manager **always** branches to ARIUXIT regardless of whether the connect attempt is from a program, the DBS utility, the preprocessors, or ISQL. You cannot disable branching. If you want to be able to bypass your accounting routine, you have to code the routine so that you can turn it on and off.

How the ARIUXIT Module Works

The resource adapter is the component of the database manager that calls ARIUXIT. In multiple user mode, the resource adapter is in the user partition; in single user mode, it is in database partition. The ARIUXIT module is called in both modes.

When the resource adapter detects any attempt to connect to an application server, it builds a parameter list for ARIUXIT, sets registers for a proper linkage, and calls ARIUXIT. It always calls ARIUXIT, even if the accounting facility is not enabled. The registers are set as follows:

Register 1	The address of the start of the parameter list for ARIUXIT. The parameter list itself is named ARIUEXI. The pointer to the parameter list points to the beginning of ARIUEXI, which is described below. The first field in ARIUEXI is an eye-catcher value.
Register 13	Points to a standard register save area.
Register 14	Contains the return address.
Register 15	Contains the entry point of the installation exit routine.

You must code ARIUXIT to save the DB2 Server for VSE registers in the area pointed to by Register 13. If ARIUXIT does not save and restore the registers, the results will be unpredictable.

The resource adapter also builds the parameter list named ARIUEXI. Table 25 shows what is in ARIUEXI.

Table 25. ARIUEXI Parameter List

Length	Description
2 words 1 word	Eye-catcher: 'ARIUEXI ' Length of ARIUEXI parameter list
1 word 1 word	Pointer to Exit Number Pointer to length of Exit Number
1 word 1 word	Pointer to Exit Global Area Pointer to length of Exit Global Area
1 word 1 word	Pointer to Exit Local User Area Pointer to length of Exit Local User Area
1 word 1 word	Pointer to Exit Unique Area Pointer to length of Exit Unique Area
2 words	Reserved
1 word 1 word	Pointer to Environment Dependent Area Pointer to length of Environment Dependent Area
1 word 1 word	Pointer to Exit Return Code Area Pointer to length of Exit Return Code Area

Each area that ARIUEXI points to is described below.

Eye-catcher and Length of List
: The resource adapter sets the eye-catcher field to 'ARIUEXI ' and the following full word to the length of the entire list. (This length includes the length of the eye-catcher field.)

Exit Number
: The exit number is always a full word. The exit number for the accounting exit is 1. The resource adapter sets the pointer to the exit number with an

address to a full word area containing a binary 1. The resource adapter sets the pointer to the length of the exit number with the address of a full word area containing a binary 4.

Exit Global Area
This area does not apply to the accounting exit. The resource adapter sets both the pointer to the global area and the pointer to the length of the global area to binary zeros.

Exit Local User Area
The local user area is 16 bytes long. It is a read/write area that lasts for the life of the user program.

- For CICS transactions, the area exists for each transaction until the transaction ends
- For batch or VSE/ICCF applications, the area exists until the end of the job step.

For each user, the resource adapter obtains the 16-byte storage area and sets it to binary zeros. The pointer to the local user area is unique for each user. On subsequent calls by the user, the resource adapter returns the same pointer; it never resets the area.

The pointer to the length of the local user area always points to a full word that contains a binary 16.

Exit Unique Area
In this area, you provide the installation-dependent accounting information. This area is also 16 bytes long. It is obtained and initialized by the resource adapter. How it is initialized depends on the environment:

- For batch/ICCF applications, it is initialized to character blanks.
- For online CICS transactions, it is initialized as follows:

  Bytes 1—4
  contain the CICS transaction ID.

  Bytes 5—12
  contain the CICS sigon ID, if available.

  Bytes 13—16
  contain the CICS terminal ID, if available. If the transaction does not have a CICS terminal ID and the user coded the ARIRCAN cancel support, the first 4 characters of the RMARUDAT data from the RMAR control block is placed into bytes 13-16. Otherwise, these bytes are blank.

  Note: If the user coded the ARIRCAN cancel support, the RMARUDAT data from the RMAR control block is placed into bytes 29-36 of the Environment Dependent area.

Reserved Area
This area is 8 bytes long and reserved. The resource adapter initializes it to binary zeros.

Environment-Dependent Area
This area is 40 bytes long. It contains information about the environment where the user is running.

Note: This area identifies environment-dependent information. Some fields apply only to VM uses of the database manager. For VSE, those fields are set to character blanks.

The resource adapter initializes the environment-dependent area as follows:

Byte 1 Character S for single user mode, or M for multiple user mode.

Byte 2 Character D for VSE.

Byte 3 Character B for batch or VSE/ICCF, or O for VSE online.

Byte 4 Character I for implicit connect, or E for explicit connect.

Bytes 5—8
Pointer to CICS transaction control area (TCA) for VSE online. For batch or VSE/ICCF, binary zeros.

Bytes 9—12
Pointer to CICS save area (CSA) for VSE online. For batch or VSE/ICCF, this field is set to binary zeros.

Bytes 13—20
Character blanks.

Bytes 21—28
CONNECT user ID for all explicit connections and all online implicit connections. Blanks for implicit connections.

Bytes 29—36
If the ARIRCAN cancel support is coded, this field is set to RMARUDAT from the RMAR control block. (For more information about the contents of this field, see the description of the Exit Unique Area on page 266.) Otherwise, this field is character blanks.

Bytes 37—40
Binary zeros (reserved).

Exit Return Code Area
The resource adapter initializes this full word area (and the pointer to it), and sets it to -1. A return code of -1 means that you do not want this exit. The length field for this area is a full word containing a binary 4. The resource adapter also ORs a X'80' to the high order byte of the pointer to the length field of the return code area. The X'80' indicates the end of the parameter list.

When you code your version of ARIUXIT, you can specify these return codes before branching back to the database manager:

-1 Means that you do not want this exit (the default). This indicates to the database manager that the exit is a no-op.

0 The function that the exit called to do a task completed successfully.

Other Any return code other than 0 or -1 causes an -815 SQLCODE to be returned to the user. (SQLERRD1 contains the return code from the exit.) You can reject a user's attempt to connect because the user has incorrect accounting information.

Figure 105 on page 268 summarizes the ARIUEXI parameter list and the areas pointed to by the list.

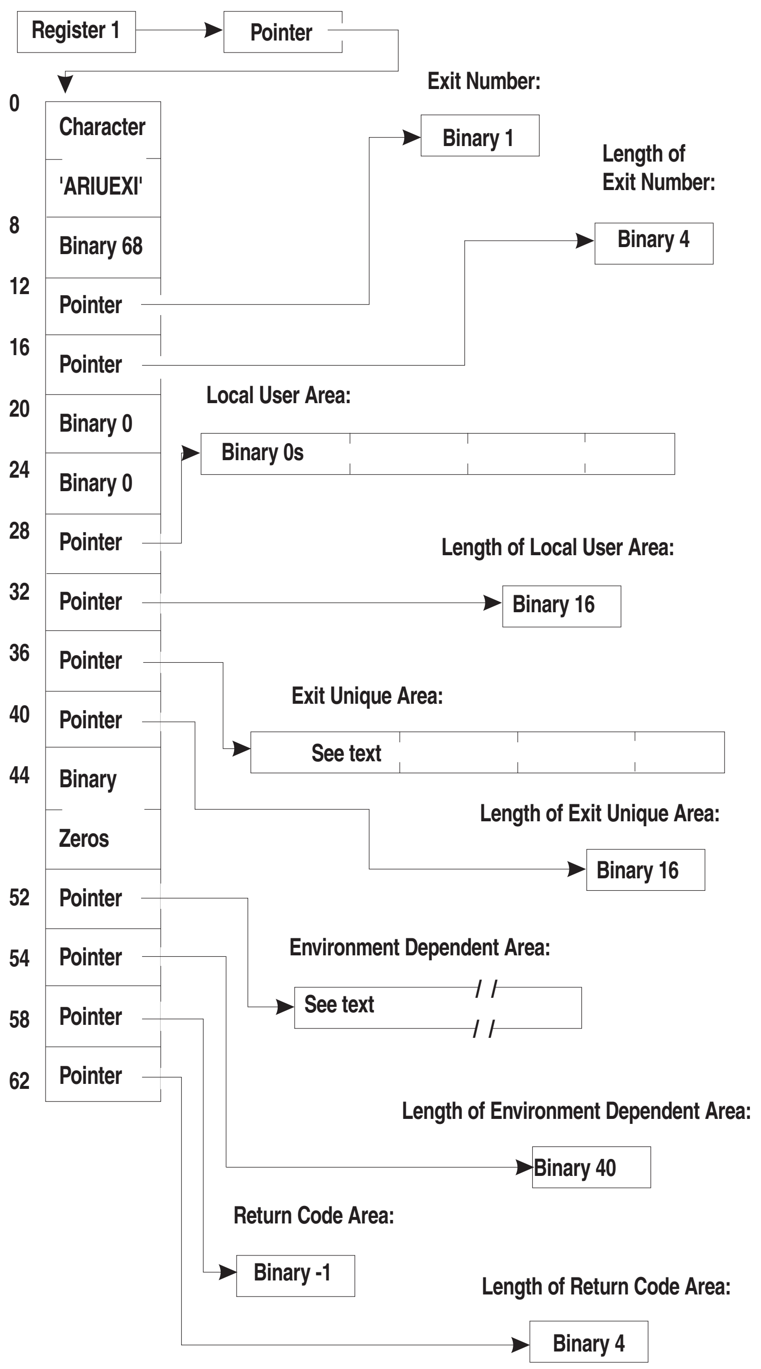


Figure 105. Summary of ARIUEXI Parameter List and Associated Areas

Coding Your Own Accounting Exit

Exit routines must always be coded in Assembler language. Your version of ARIUXIT (and any of the modules it calls) must not use any DB2 Server for VSE function. In an online environment, imbedded CICS commands (EXEC CICS) are not allowed.

You can link-edit your accounting exit to run in AMODE 24 or 31 by using the AMODE parameter in the PARM field of the EXEC LNKEDT statement. Your accounting exit will be loaded below 16 megabytes and given control according to its AMODE. If the AMODEs of the accounting exit and the database manager differ, the database manager switches AMODEs before transferring control to the accounting exit. For CICS/VSE applications, you MUST link-edit your accounting exit to run in AMODE 31.

Figure 106 shows the ARIUXIT module that is included with the database manager. This sample exit is supplied as an A-type source member named ARIUXIT. Note that the Exit Return Code Area is set to -1, which means that you are not interested.

```
         TITLE 'ARIUXIT'
***********************************************************************
* ARIUXIT USER EXIT ROUTER ROUTINE                                    *
*     REGISTER ASSUMPTIONS:                                           *
*         R1  -> PARMLIST                                             *
*         R13 -> SAVE AREA                                            *
*         R14 -> RETURN ADDRESS                                       *
*         R15 -> ENTRY POINT                                          *
*                                                                     *
* ALTHOUGH PROVIDED IN A GENERAL INTERFACE LIBRARY, ARIUXIT IS NOT TO *
* BE USED AS A GENERAL PROGRAMMING INTERFACE.  REFER TO PRODUCT       *
* DOCUMENTATION TO DETERMINE INTENDED USAGE.                          *
*                                                                     *
***********************************************************************
         SPACE 5
ARIUXIT  CSECT ,
ARIUXIT  AMODE ANY
ARIUXIT  RMODE 24
         DS    0H
         USING *,R15                GET ADDRESSABILITY
         B     PROLOG
         DC    CL8'ARIUXIT '        EYECATCHER
*
PROLOG   EQU   *
         STM   R14,R12,12(R13)      SAVE CALLER'S REGISTERS
         DROP  R15
         BALR  R12,0                R12 IS BASE REGISTER
*
PSTART   EQU   *
         USING PSTART,R12           GET ADDRESSABILITY FOR ROUTINE
         ST    R13,UXSAVE+4         STORE BACKWARD POINTER
         LA    R9,UXSAVE            ADDRESS OF SAVE AREA
         ST    R9,UXSAVE+8          STORE FORWARD POINTER
         LR    R13,R9               R13 POINTS TO NEW SAVE AREA
         L     R1,0(,R1)            GET POINTER TO PLIST
         USING PLIST,R1             GET ADDRESSABILITY TO PLIST DSECT
*
*  Insert your own code here
*  (and change the return code as appropriate).
*
```

Figure 106. IBM-Supplied Version of ARIUXIT (Part 1 of 3)

```
         L     R2,PLRETCD          GET PTR TO EXIT RETURN CODE AREA
         L     R3,NEG1RC           LOAD NOOP RET CODE (NEGATIVE ONE)
*
         ST    R3,0(,R2)           STORE RET CODE INTO EXIT RC AREA
*
         L     R13,UXSAVE+4        GET BACKWARD POINTER
         LM    R14,R12,12(R13)     RESTORE CALLER'S REGISTERS
         BR    R14                 RETURN TO CALLER
*
END      EQU   *
         EJECT
***********************************************************************
*
*        DECLARES FOR ARIUXIT ROUTER
*
***********************************************************************
         SPACE 5
UXSAVE   DC    18F'0'         SAVE AREA FOR CALLER'S REGISTERS
NEG1RC   DC    F'-1'          NEGATIVE ONE RETURN CODE (NO-OP)
         SPACE 2
R0       EQU   0              REGISTERS EQUATES
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         EJECT
```

Figure 106. IBM-Supplied Version of ARIUXIT (Part 2 of 3)

```
**********************************************************************
*
*        DSECT FOR ARIUEXI PARAMETER LIST INTERFACE TO ARIUXIT ROUTER
*
**********************************************************************
         SPACE 5
         DS    0D
PLIST    DSECT
PLICTCH  DS    CL8            EYECATCHER
PLILENG  DS    F              LENGTH OF PLIST (INCLUDING EYECATCHER)
PLEXNUM  DS    F              PTR TO EXIT NUMBER
PLLXNUM  DS    F              PTR TO LENGTH OF EXIT NUMBER
PLGLOBA  DS    F              PTR TO EXIT GLOBAL AREA
PLLGLOB  DS    F              PTR TO LENGTH OF EXIT GLOBAL AREA
PLUSERF  DS    F              PTR TO EXIT LOCAL USER AREA
PLLUSER  DS    F              PTR TO LENGTH OF EXIT LOCAL USER AREA
PLEUNIQ  DS    F              PTR TO EXIT UNIQUE AREA
PLLUNIQ  DS    F              PTR TO LENGTH OF EXIT UNIQUE AREA
         DS    CL8            RESERVED
PLEDEPA  DS    F              PTR TO ENVIRONMENT DEPENDENT AREA
PLLDEPA  DS    F              PTR TO LENGTH OF ENVIRONMENT DEP AREA
PLRETCD  DS    F              PTR TO EXIT RETURN CODE AREA
PLLRETC  DS    F              PTR TO LENGTH OF EXIT RETURN CODE AREA
*
         END
```

Figure 106. IBM-Supplied Version of ARIUXIT (Part 3 of 3)

Figure 107 shows a simple example of a user version of ARIUXIT. In this example, the string HERE IS USERDATA is moved into the exit unique area, and the exit return code area is set to 0.

```
          TITLE 'ARIUXIT'
***********************************************************************
* ARIUXIT USER EXIT ROUTER ROUTINE                                    *
*     REGISTER ASSUMPTIONS:                                           *
*         R1  -> PARMLIST                                             *
*         R13 -> SAVE AREA                                            *
*         R14 -> RETURN ADDRESS                                       *
*         R15 -> ENTRY POINT                                          *
*                                                                     *
* ALTHOUGH PROVIDED IN A GENERAL INTERFACE LIBRARY, ARIUXIT IS NOT TO *
* BE USED AS A GENERAL PROGRAMMING INTERFACE.  REFER TO PRODUCT       *
* DOCUMENTATION TO DETERMINE INTENDED USAGE.                          *
*                                                                     *
***********************************************************************
         SPACE 5
ARIUXIT  CSECT ,
ARIUXIT  AMODE ANY
ARIUXIT  RMODE 24
         DS    0H
         USING *,R15               GET ADDRESSABILITY
         B     PROLOG
         DC    CL8'ARIUXIT '       EYECATCHER
*
PROLOG   EQU   *
         STM   R14,R12,12(R13)     SAVE CALLER'S REGISTERS
         DROP  R15
         BALR  R12,0               R12 IS BASE REGISTER
*
PSTART   EQU   *
         USING PSTART,R12          GET ADDRESSABILITY FOR ROUTINE
         ST    R13,UXSAVE+4        STORE BACKWARD POINTER
         LA    R9,UXSAVE           ADDRESS OF SAVE AREA
         ST    R9,UXSAVE+8         STORE FORWARD POINTER
         LR    R13,R9              R13 POINTS TO NEW SAVE AREA
         L     R1,0(,R1)           GET POINTER TO PLIST
         USING PLIST,R1            GET ADDRESSABILITY TO PLIST DSECT
```

Figure 107. Sample User Version of ARIUXIT (Part 1 of 3)

```
*
* Here you would place code that gets and verifies your
* user-dependent data.  The following code shows moving the data
* into the Exit Unique Area.
*
* Make sure you check the Exit Number word.  If the Exit Number value
* is not a binary 1, you should set the Exit Return Code word to binary
* -1 (NEG1RC) and return to the database manager.
*
         L     R2,PLEUNIQ          GET PTR TO EXIT UNIQUE AREA
         MVC   0(16,R2),USERDATA   MOVE 16 BYTES OF USER DATA
         L     R2,PLRETCD          GET PTR TO EXIT RETURN CODE AREA
         L     R3,ZEROS            SET ZERO RETURN CODE
         ST    R3,0(,R2)           STORE RET CODE INTO EXIT RC AREA
*
EXIT     EQU   *                   RETURN TO THE DATABASE MANAGER
         L     R13,UXSAVE+4        GET BACKWARD POINTER
         LM    R14,R12,12(R13)     RESTORE CALLER'S REGISTERS
         BR    R14                 RETURN TO CALLER
*
END      EQU   *
         EJECT

***********************************************************************
*
*        DECLARES FOR ARIUXIT
*
***********************************************************************
         SPACE 5
UXSAVE   DC    18F'0'         SAVE AREA FOR CALLER'S REGISTERS
ZEROS    DC    F'0'           ZERO RETURN CODE
NEG1RC   DC    F'-1'          NEGATIVE RETURN CODE (NO-OP)
USERDATA DC    CL16'HERE IS USERDATA'
         SPACE 2
R0       EQU   0              REGISTERS EQUATES
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         EJECT
```

Figure 107. Sample User Version of ARIUXIT (Part 2 of 3)

```
***********************************************************************
*
*        DSECT FOR ARIUEXI PARAMETER LIST INTERFACE TO ARIUXIT
*
***********************************************************************
         SPACE 5
         DS    0D
PLIST    DSECT
PLICTCH  DS    CL8            EYE-CATCHER
PLILENG  DS    F              LENGTH OF PLIST (INCLUDING EYE-CATCHER)
PLEXNUM  DS    F              PTR TO EXIT NUMBER
PLLXNUM  DS    F              PTR TO LENGTH OF EXIT NUMBER
PLGLOBA  DS    F              PTR TO EXIT GLOBAL AREA
PLLGLOB  DS    F              PTR TO LENGTH OF EXIT GLOBAL AREA
PLUSERF  DS    F              PTR TO EXIT LOCAL USER AREA
PLLUSER  DS    F              PTR TO LENGTH OF EXIT LOCAL USER AREA
PLEUNIQ  DS    F              PTR TO EXIT UNIQUE AREA
PLLUNIQ  DS    F              PTR TO LENGTH OF EXIT UNIQUE AREA
         DS    CL8            RESERVED
PLEDEPA  DS    F              PTR TO ENVIRONMENT DEPENDENT AREA
PLLDEPA  DS    F              PTR TO LENGTH OF ENVIRONMENT DEP AREA
PLRETCD  DS    F              PTR TO EXIT RETURN CODE AREA
PLLRETC  DS    F              PTR TO LENGTH OF EXIT RETURN CODE AREA
*
         END
```

Figure 107. Sample User Version of ARIUXIT (Part 3 of 3)

After the program is coded, assemble it as you would any other program.

Installing Your Version of ARIUXIT

After assembling your program, you must catalog the ARIUXIT OBJ file into your private sublibrary. (Your assembled version of ARIUXIT must be named ARIUXIT OBJ.) Then link-edit the DB2 Server for VSE component DSC; then stop the application server in order for the change to take effect.

The DSC component, which is described in the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual, contains the IBM-supplied version of ARIUXIT. The link book name for DSC is ARISLKMD. When doing the link-edit, specify your private sublibrary ahead of the DB2 Server for VSE sublibrary on the LIBDEF statement defining the search order. The link-edit will then replace the IBM-supplied version with your version.

An example of job control to install a user version of ARIUXIT is shown in Figure 108 on page 275. Here, the ARIUXIT OBJ would be installed in a sublibrary called LIB.SQL, and the DSC component would be replaced in PRD2.DB2730.

```
// JOB INSTALL USER EXIT
// LIBDEF *,SEARCH=(LIB.SQL,PRD2.DB2730),CATALOG=PRD2.DB2730
* ***************************************************************
* INSTALL ARIUXIT USER EXIT ROUTINE                             *
* ***************************************************************
// OPTION CATAL
   INCLUDE ARISLKMD
// EXEC LNKEDT,PARM='MSHP,AMODE=xxx,RMODE=24'
/*
/&
```

Figure 108. Example Job Control to Install ARIUXIT in the Production Library

AMODE (addressing mode) is the attribute of the entry point of the loaded phase, and must be one of the following:

24 The exit is invoked in AMODE 24

31 The exit is invoked in AMODE 31

ANY The exit is invoked in the caller's addressing mode

Note: The accounting exit must always be loaded below 16 megabytes (RMODE 24). If you are running DRDA online CICS transactions, the accounting exit cannot be linkedited with AMODE 24.

Service Considerations for ARIUXIT

The dummy version of ARIUXIT is not serviceable; other portions of the DSC component, however, are serviceable. If service is applied to any portion of DSC, it is link-edited again. If you have coded your own version of ARIUXIT and completed the previous steps, your version of ARIUXIT will be included in the DSC component.

Defining Your Own Datetime Format

The database manager supports many datetime formats. This section describes the datetime formats and how you can add your own by coding your own exit.

Datetime Formats

The database manager supports DATE, TIME, and TIMESTAMP data types and operations. You can enter a date or a time using many different formats.

Dates can be entered in any of the formats shown in Table 26.

Table 26. Date Formats

Format Name	Abbreviation	Date Format	Example
International Standards Organization	ISO	*yyyy-mm-dd*	1993-12-31
IBM USA standard	USA	*mm/dd/yyyy*	12/31/1993
IBM European standard	EUR	*dd.mm.yyyy*	31.12.1993
Japanese Industrial Standard Christian Era	JIS	*yyyy-mm-dd*	1993-12-31
Site-defined	LOCAL	Any site-defined form	—

Times can be entered in any of the formats shown in Table 27.

Table 27. Time Formats

Format Name	Abbreviation	Time Format	Example
International Standards Organization	ISO	*hh.mm*[.*ss*]	13.30.05
IBM USA standard	USA	*hh:mm* AM or PM	1:30 PM
IBM European standard	EUR	*hh.mm*[.*ss*]	13.30.05
Japanese Industrial Standard Christian Era	JIS	*hh:mm*[:*ss*]	13:30:05
Site-defined	LOCAL	Any site-defined form	—

To define the LOCAL format, you have to code your own date or time exit. For information about coding your own datetime exit, see "Coding Your Own Datetime Exit" on page 279.

Default Output Format

When the database manager is installed, the default date and time formats are both ISO. To change them, you must change the entry in the SYSTEM.SYSOPTIONS table. You must have DBA authority to do this.

For example, to specify that the date output format is USA, enter:

```
UPDATE SYSTEM.SYSOPTIONS -
  SET VALUE='USA' WHERE SQLOPTION='DATE'
```

Similarly, to specify that the time output format is JIS, you enter:

```
UPDATE SYSTEM.SYSOPTIONS -
  SET VALUE='JIS' WHERE SQLOPTION='TIME'
```

Alternatively, you can update the SYSTEM.SYSOPTIONS table by modifying the IBM-supplied ARISDTM member to specify your datetime defaults, then start the DBS utility, specifying the ARISDTM member as the control file.

How Datetime Exits Work

Two datetime installation replaceable exits are provided to allow you to convert datetime values in any installation-defined format into ISO format, or from ISO format into any installation-defined format. These exits which are link-edited into the exit router component ARIXSXR, are called ARIUXDT and ARIUXTM for date and time, respectively.

When the database manager is installed, ARIXSXR is loaded and addressability to the exits is set.

The entries in the SYSTEM.SYSOPTIONS catalog table are used by the database manager to determine the default datetime format for output.

If SYSTEM.SYSOPTIONS indicates that local datetime exits are present, the exits are called during SQL statement processing when conversion between internal and external forms is required.

The product-supplied exits return a -1 return code, meaning the exits have not been replaced by the user exits. If a user program issues an SQL statement that calls the exits, SQLCODE -185 is returned. Therefore, if the user is to replace the exits, the -1 return code must **not** be used.

When Date and Time Exits are Called (Exit Points)

If a program has been preprocessed with the LOCAL format, or if the installation default is LOCAL, then the datetime exits are called before any interpretation of the datetime data values. Otherwise, the database manager attempts to interpret the datetime data values first. In this situation, it calls the local exit only if it does not recognize the datetime value.

The datetime exits are called at the following times:

- When you convert the external form to an internal form:
  - When datetime data is entered by INSERT or UPDATE statements, or by the DATALOAD commands of the DBS utility, or by the INPUT command of ISQL.
  - When a constant or host variable is compared with a DATE or TIME column. The constant can be converted during preprocessing time.
  - When the DATE or TIME scalar functions are used with a string representation of a date or time.

  The exit should then convert the installation-defined format into ISO format. The ISO format is then validated and converted into an internal format to be entered into the column or used in comparisons. If the column is a key column for an index, the index entry is made in an internal format.
- When you convert the internal form to an external form:
  - When data is retrieved from the column by SELECT or FETCH statements, or by the DATAUNLOAD commands in the DBS utility, and the default format is local.
  - When the CHAR scalar function is used with the LOCAL format specification.

  At this point, the exit should convert the value from ISO format into installation-defined format; then the database manager returns the converted value. In this situation, the exit is called after any edit routine or sort.

When the exits are called, the registers are set as follows:

Register 0	Undefined.
Register 1	Points to a pointer to the parameter list for ARIUXDT (or ARIUXTM). The format of the parameter list is discussed below. The first field in it is an eye-catcher value.
Register 2—12	Undefined.
Register 13	Points to a standard register save area.
Register 14	Contains the return address.
Register 15	Contains the entry point of the user installation routine.

Registers 2—13 must be saved and restored by the exit. If this is not done, the results will be unpredictable.

Table 28 shows what is in the parameter list used by the date and time exits (see Register 1).

Table 28. Parameter List Used by Date and Time Exits

Length	Description
2 words 1 word	Eye-catcher: ARIUXDT or ARIUXTM Length of parameter list
1 word 1 word	Pointer to Function Number Pointer to length of Function Number
1 word 1 word	Pointer to Exit Global Area Pointer to length of Exit Global Area
1 word 1 word	Pointer to ISO Datetime Area Pointer to length of ISO Datetime Area
1 word 1 word	Pointer to LOCAL Datetime Area Pointer to length of LOCAL Datetime Area
1 word 1 word	Pointer to User Work Area Pointer to length of User Work Area
1 word 1 word	Pointer to Environment Dependent Area Pointer to length of Environment Dependent Area
1 word 1 word	Pointer to Exit Return Code Area Pointer to length of Exit Return Code Area

Each area in the parameter list is described below.

- The Eye-catcher and Length of list is initialized by the database manager.
- The Function Number is a full word number describing the function to be performed, as follows:

Number	Function
	DATE Functions:
00000004	Convert DATE from LOCAL format to ISO format.
00000008	Convert DATE from ISO format to Installation format.
	TIME Functions:
00000004	Convert TIME from LOCAL format to ISO format.
00000008	Convert TIME from ISO format to Installation format.

- The EXIT Global Area is not used. Both values are set to zero.
- The length of the ISO Date and Time Areas are 10 bytes and 8 bytes, respectively.
- The length of the LOCAL Date and Time Areas are as defined in the SYSTEM.SYSOPTIONS catalog table. The pointer to the length of the area points to a fullword that contains the value in this table.
- The User Work Area is a 512-byte area.
- The Environment Dependent Area is a 40-byte area. For the datetime exits, only byte 2 is used. It contains D for VSE, and V for VM.
- The Exit Return Code Area is a full word to be set by the exit to the return code.

  The possible return codes are:

  -1 The exit supplied by the database manager has not been replaced by a user exit. The database manager then sets SQLCODE to -185.

  0 The function has been performed.

  4 Invalid date or time value. The database manager then sets SQLCODE to -181.

8 Date or time value not in valid format. The database manager then sets SQLCODE to -180.

Other Error in exit. The function number of the exit will be stored in SQLERRD5, and the return code in SQLERRD1. The database manager then sets SQLCODE to -816.

The exit name, function code and return code are set up as message tokens in SQLERRM; they are used when the message associated with SQLCODE -816 is displayed, for example, by the DBS utility and ISQL.

If a program has been preprocessed with the LOCAL format, or if the installation default is LOCAL, then the database manager evaluates the output of the datetime exit if the return code is either `0` or 8. Otherwise, the output is evaluated only if the return code is `0`.

Coding Your Own Datetime Exit

User-coded exits must conform to the following:

- The installation replaceable exits must be coded in Assembler language.
- The exits must be reentrant; they must save registers at entry and restore them before exit.
- The exits (and any of the modules they call) must not use any DB2 Server for VSE facilities.
- The exits must not use the return code -1.
- When formatting ISO datetime to LOCAL datetime, the user is responsible for formatting the full buffer (the number of bytes equal to the length of the local datetime as defined in the SYSTEM.SYSOPTIONS catalog table).
- The exits must support 31-bit addressing (AMODE 31) and be loaded below 16 megabytes (RMODE 24)

In an online environment, imbedded CICS commands (EXEC CICS) are not allowed.

Figure 109 on page 280 shows the IBM-supplied ARIUXDT module, which is an A-type source member. You need to modify this source code to support your local date format requirements.

```
         TITLE ' ARIUXDT'
***********************************************************************
* ARIUXDT USER DATE CONVERSION ROUTINE                                *
*     REGISTER ASSUMPTIONS:                                           *
*         R1  -> PARMLIST                                             *
*         R13 -> SAVE AREA                                            *
*         R14 -> RETURN ADDRESS                                       *
*         R15 -> ENTRY POINT                                          *
*                                                                     *
* ALTHOUGH PROVIDED IN A GENERAL INTERFACE LIBRARY, ARIUXIT IS NOT TO *
* BE USED AS A GENERAL PROGRAMMING INTERFACE.  REFER TO PRODUCT       *
* DOCUMENTATION TO DETERMINE INTENDED USAGE.                          *
***********************************************************************
ARIUXDT  CSECT ,
ARIUXDT  AMODE 31
ARIUXDT  RMODE 24
         USING *,R15                ESTABLISH TEMP ADDRESSABILITY
         B     PROLOG               BRANCH TO START OF PROGRAM
         DC    C'ARIUXDT'
         DROP  R15                  DROP R15 AND USE OWN ADDRESSABIL-
*                                   ITY
PROLOG   STM   R14,R12,12(R13)      SAVE REGS IN CALLER'S AREA
         LR    R12,R15              SAVE BASE REGISTER
PSTART   EQU   ARIUXDT              START OF PROGRAM
         USING PSTART,R12           SET UP BASE REGISTER
         L     R1,0(R1)             POINT TO THE PARAMETER LIST
         USING PARMLIST,R1          ADDRESSABILITY FOR INPUT PARMS
         L     R2,FNPTR             POINT TO FUNCTION TYPE
***********************************************************************
*  M A I N L I N E
***********************************************************************
MAINLINE DS    0H                   START OF CODE
         SPACE
         SR    R15,R15              INITIALIZE RETURN CODE TO ZERO
***********************************************************************
* HERE YOU WOULD PLACE CODE THAT GETS AND VERIFIES YOUR
* INPUT DATE AND CONVERTS IT TO EITHER TO LOCAL FORMAT OR ISO FORMAT
* A RETURN CODE OF -1 MEANS AN EXIT IS NOT PROVIDED
* A RETURN CODE OF 0 MEANS CONVERSION WAS SUCCESSFUL
* A RETURN CODE OF 4 MEANS THAT THE DATE VALUE WAS OUT OF RANGE
* A RETURN CODE OF 8 MEANS THAT THE DATE WAS INVALID
***********************************************************************
         BCTR  R15,R0               EXIT NOT PROVIDED
         B     RETURN               CONVERSION COMPLETE
```

Figure 109. IBM-Supplied Version of ARIUXDT (Part 1 of 2)

```
***********************************************************************
*  RETURN TO CALLER
***********************************************************************
RETURN   DS    0H                     RETURN POINT
         L     R2,RETPTR              LOAD RETCODE PTR
         ST    R15,0(R2)              STORE EXIT RETURN CODE
         L     R14,12(,R13)           RESTORE R14
         LM    R0,R12,20(R13)         RESTORE REST OF CALLER'S REGS
         BR    R14                    RETURN TO CALLER
         EJECT
PARMLIST DSECT ,                      INPUT PARAMETER LIST
EYECATCH DS    CL8                    EYECATCHER
PLEN     DS    F                      LENGTH OF PARAMETER LIST
FNPTR    DS    AL4                    POINTER TO FUNCTION TYPE
FNLENP   DS    AL4                    LENGTH OF FUNCTION TYPE
GLBPTR   DS    AL4                    POINTER TO GLOBAL EXIT AREA
GLBLENP  DS    AL4                    LENGTH OF GLOBAL EXIT AREA
ISOPTR   DS    AL4                    POINTER TO ISO DATETIME AREA
ISOLENP  DS    AL4                    LENGTH OF ISO DATETIME AREA
LOCPTR   DS    AL4                    POINTER TO LOCAL DATETIME AREA
LOCLENP  DS    AL4                    LENGTH OF LOCAL DATETIME AREA
WORKPTR  DS    AL4                    POINTER TO USER WORK AREA
WORKLENP DS    AL4                    LENGTH OF USER WORK AREA
ENVPTR   DS    AL4                    POINTER TO ENVIR. DEPENDANT AREA
ENVLENP  DS    AL4                    LENGTH OF ENVIR. DEPENDANT AREA
RETPTR   DS    AL4                    POINTER TO RETURN CODE AREA
RETLENP  DS    AL4                    LENGTH OF RETURN CODE AREA
         EJECT
ARIUXDT  CSECT ,
R0       EQU   00                     EQUATES FOR REGISTERS 0-15
R1       EQU   01
R2       EQU   02
R3       EQU   03
R4       EQU   04
R5       EQU   05
R6       EQU   06
R7       EQU   07
R8       EQU   08
R9       EQU   09
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END   ARIUXDT
```

Figure 109. IBM-Supplied Version of ARIUXDT (Part 2 of 2)

Figure 110 on page 282 shows the IBM-supplied ARIUXTM module. This module is an A-type source member named ARIUXTM. You can modify this source code to support your local time format requirements.

```
         TITLE ' ARIUXTM'

***********************************************************************
* ARIUXTM USER TIME CONVERSION ROUTINE                                *
*     REGISTER ASSUMPTIONS:                                           *
*         R1  -> PARMLIST                                             *
*         R13 -> SAVE AREA                                            *
*         R14 -> RETURN ADDRESS                                       *
*         R15 -> ENTRY POINT                                          *
*                                                                     *
* ALTHOUGH PROVIDED IN A GENERAL INTERFACE LIBRARY, ARIUXIT IS NOT TO *
* BE USED AS A GENERAL PROGRAMMING INTERFACE.  REFER TO PRODUCT       *
* DOCUMENTATION TO DETERMINE INTENDED USAGE.                          *
***********************************************************************
ARIUXTM  CSECT ,
ARIUXTM  AMODE 31
ARIUXTM  RMODE ANY
         USING *,R15                   ESTABLISH TEMP ADDRESSABILITY
         B     PROLOG                  BRANCH TO START OF PROGRAM
         DC    C'ARIUXTM'
         DROP  R15                     DROP R15 AND USE OWN ADDRESSABIL-
*                                      ITY
PROLOG   STM   R14,R12,12(R13)         SAVE REGS IN CALLER'S AREA
         LR    R12,R15                 SAVE BASE REGISTER
PSTART   EQU   ARIUXTM                 START OF PROGRAM
         USING PSTART,R12              SET UP BASE REGISTER
         L     R1,0(R1)                POINT TO PARAMETER LIST
         USING PARMLIST,R1             ADDRESSABILITY FOR INPUT PARMS
         L     R2,FNPTR                POINT TO FUNCTION TYPE
***********************************************************************
*  M A I N L I N E
***********************************************************************
MAINLINE DS    0H                      START OF CODE
         SPACE
         SR    R15,R15                 INITIALIZE RETURN CODE TO ZERO
***********************************************************************
* HERE YOU WOULD PLACE CODE THAT GETS AND VERIFIES YOUR
* INPUT TIME AND CONVERTS IT TO EITHER TO LOCAL FORMAT OR ISO FORMAT
* A RETURN CODE OF -1 MEANS AN EXIT IS NOT PROVIDED
* A RETURN CODE OF 0 MEANS CONVERSION WAS SUCCESSFUL
* A RETURN CODE OF 4 MEANS THAT THE TIME VALUE WAS OUT OF RANGE
* A RETURN CODE OF 8 MEANS THAT THE TIME WAS INVALID
***********************************************************************
         BCTR  R15,R0                  EXIT NOT PROVIDED
         B     RETURN                  CONVERSION COMPLETE
```

Figure 110. IBM-Supplied Version of ARIUXTM (Part 1 of 2)

```
***********************************************************************
*  RETURN TO CALLER
***********************************************************************
RETURN   DS    0H                     RETURN POINT
         L     R2,RETPTR              LOAD RETCODE PTR
         ST    R15,0(R2)              STORE EXIT RETURN CODE
         L     R14,12(,R13)           RESTORE R14
         LM    R0,R12,20(R13)         RESTORE REST OF CALLER'S REGS
         BR    R14                    RETURN TO CALLER
         EJECT
PARMLIST DSECT ,                      INPUT PARAMETER LIST
EYECATCH DS    CL8                    EYECATCHER
PLEN     DS    F                      LENGTH OF PARAMETER LIST
FNPTR    DS    AL4                    POINTER TO FUNCTION TYPE
FNLENP   DS    AL4                    LENGTH OF FUNCTION TYPE
GLBPTR   DS    AL4                    POINTER TO GLOBAL EXIT AREA
GLBLENP  DS    AL4                    LENGTH OF GLOBAL EXIT AREA
ISOPTR   DS    AL4                    POINTER TO ISO DATETIME AREA
ISOLENP  DS    AL4                    LENGTH OF ISO DATETIME AREA
LOCPTR   DS    AL4                    POINTER TO LOCAL DATETIME AREA
LOCLENP  DS    AL4                    LENGTH OF LOCAL DATETIME AREA
WORKPTR  DS    AL4                    POINTER TO USER WORK AREA
WORLENP  DS    AL4                    LENGTH OF USER WORK AREA
ENVPTR   DS    AL4                    POINTER TO ENVIR. DEPENDANT AREA
ENVLENP  DS    AL4                    LENGTH OF ENVIR. DEPENDANT AREA
RETPTR   DS    AL4                    POINTER TO RETURN CODE AREA
RETLENP  DS    AL4                    LENGTH OF RETURN CODE AREA
         EJECT
ARIUXTM  CSECT ,
R0       EQU   00                     EQUATES FOR REGISTERS 0-15
R1       EQU   01
R2       EQU   02
R3       EQU   03
R4       EQU   04
R5       EQU   05
R6       EQU   06
R7       EQU   07
R8       EQU   08
R9       EQU   09
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END   ARIUXTM
```

Figure 110. IBM-Supplied Version of ARIUXTM (Part 2 of 2)

After the program is coded, assemble it as you would any other program.

Installing Your Version of ARIUXDT or ARIUXTM

After assembling your program, you must catalog the ARIUXDT TEXT (or ARIUXTM TEXT) file into your private sublibrary. (Your assembled version of ARIUXDT or ARIUXTM must be named ARIUXDT TEXT or ARIUXTM TEXT.) Then link-edit the exit router component ARIXSXR.

The ARIXSXR component contains the IBM-supplied version of ARIUXDT (or ARIUXTM). The link book name is ARISLKXD. When doing the link-edit, specify your private sublibrary ahead of the DB2 Server for VSE sublibrary on the LIBDEF statement that defines the search order. The link-edit will then replace the IBM-supplied version with your version.

An example of job control to install a user version of ARIUXDT or ARIUXTM is shown in Figure 111. Here, it is assumed that the ARIUXDT OBJ (or ARIUXTM OBJ) file is in a sublibrary called LIB.USER, and the ARIXSXR component will be replaced in PRD2.DB2730.

```
// JOB INSTALL USER
// LIBDEF *,SEARCH=(LIB.USER,PRD2.DB2730),CATALOG=PRD2.DB2730
* ***************************************************************
* INSTALL ARIUXIT USER EXIT ROUTINE                             *
* ***************************************************************
// OPTION CATAL
   INCLUDE ARISLKXD
// EXEC LNKEDT,PARM='MSHP'
/*
/&
```

Figure 111. Example Job Control to Install ARIUXDT or ARIUXTM

Updating the SYSTEM.SYSOPTIONS Catalog Table

You need to update the SYSTEM.SYSOPTIONS catalog table to specify the length of your local datetime format.

If you installed a local date or time format, you can update the local date or time length by using the database manager. For example, if the length of your local date format is 10 bytes, enter:

```
UPDATE SYSTEM.SYSOPTIONS -
  SET VALUE = '10' -
  WHERE SQLOPTION = 'LDATELEN'
```

The local date length specified must be greater than 9 and less than 255.

If the length of your local time format is 8 bytes, enter:

```
UPDATE SYSTEM.SYSOPTIONS -
  SET VALUE = '8' -
  WHERE SQLOPTION = 'LTIMELEN'
```

The local time length specified must be greater than 7 and less than 255.

The changes will be in effect the next time the application server is started.

You can also update the SYSTEM.SYSOPTIONS table by modifying the IBM-supplied ARISDTM member to specify your datetime defaults, then call the DBS utility, specifying the ARISDTM member as the control file.

Coding Your Own TRANSPROC Exit

General-Use Programming Interface

The TRANSPROC exit is a General-Use programming interface. General-Use programming interface is defined in “Programming Interface Information” on page 425.

The TRANSPROC exit is used for DBCS conversion. The database manager converts DBCS characters from one DBCS CCSID to another by using the value specified in the TRANSPROC column of the SYSTEM.SYSSTRINGS catalog table. This conversion can be performed when the CCSID of the source and the target are both mixed or are both graphic; that is, the TRANSTYPE column of SYSTEM.SYSSTRINGS has a value of 'PM', 'MM', or 'GG'.

The TRANSPROC exit is also used

- for EUC conversion, to convert MBCS data to mixed data. In EUC conversions, the TRANSTYPE column is either 'PM', or 'GG'.
- to convert Unicode data to CCSIDs that are supported on the database manager. For Unicode to host conversions, the TRANSTYPE column is one of 'US', 'UM' 'UG', or 'UI'.

If you have created your own DBCS CCSIDs, you must create your own conversion routine. To do so:

1. Compile, link-edit and GENMOD your routine to create a MODULE file, and store the module on the production disk.
2. Insert the name of the phase in the TRANSPROC column of the row for which you want either mixed-to-mixed or graphic-to-graphic conversion. (For example, you could create and run a DBSU job to perform this task.)
3. Stop the application server.
4. Run the job control program ARISCNVD to regenerate the CCSID-related phases. See the *DB2 Server for VSE Program Directory* manual for more information on the job control program ARISCNVD.
5. Restart the application server.

The interface between the database manager and a DBCS conversion routine supplied by a user must conform to the following:

- Register conventions:
  - Register `0` is undefined.
  - Register 1 contains the address of the control block that contains the parameters.
  - Registers 2—12 are undefined.
  - Register 13 contains the address of a standard register save area.
  - Register 14 contains the return address.
  - Register 15 contains the address of the user routine.

  Registers 2 to 13 must be saved and restored by the routine. If this is not done, the results are unpredictable.
- Parameter list, which is in the following form:
  - Address of the data to be converted (4 bytes)
  - Address of the target for the converted data (4 bytes)
  - Size of the source data (2 bytes)
  - Size of the target area (2 bytes)
  - Return code of the routine (4 bytes).

The TRANSPROC must support 31-bit addressing.

The database manager ensures that the size of the target area is at least as large as that of the source data, and that the size of the source data is always an even number. The routine supplied by the user should only convert the source data and put it in the target area. The database manager should do all other operations, such as padding the target area after data conversion is complete. You should also ensure that the routine supplies a nonzero return code if the conversion fails. The routine that you code should not have the same name as any of the defaults supplied by the database manager for the TRANSPROC column. Figure 112 shows the shell for a TRANSPROC routine.

```
TITLE 'DBCSCONV' ************************************************************************
* DBCSCONV USER DBCS CONVERSION ROUTINE
*     REGISTER ASSUMPTIONS:
*         R1  -> PARMLIST
*         R13 -> SAVE AREA
*         R14 -> RETURN ADDRESS
*         R15 -> ENTRY POINT
*
* THIS ROUTINE SHOWS THE INTERFACE TO  DB2 Server for VSE
***********************************************************************
DBCSCONV CSECT ,
DBCSCONV AMODE 31 DBCSCONV RMODE ANY
         USING *,R15                   ESTABLISH TEMP ADDRESSABILITY
         B     PROLOG                  BRANCH TO START OF PROGRAM
         DC    C'DBCSCONV'
         DROP  R15                     DROP R15 AND USE OWN ADDRESSABILITY
PROLOG   STM   R14,R12,12(R13)         SAVE REGS IN CALLER'S AREA
         LR    R12,R15                 SAVE BASE REGISTER
PSTART   EQU   DBCSCONV                START OF PROGRAM
         USING PSTART,R12              SET UP BASE REGISTER
         L     R1,0(R1)                POINT TO PARAMETER LIST
         USING PARMLIST,R1             ADDRESSABILITY FOR INPUT PARMS
```

Figure 112. TRANSPROC Shell (Part 1 of 2)

```
***********************************************************************
*  M A I N L I N E
***********************************************************************
MAINLINE DS    0H                     START OF CODE
         SPACE
***********************************************************************
* HERE YOU PLACE THE CODE THAT CONVERTS THE INPUT DBCS STRING AND
* PLACES THE CONVERTED STRING IN THE TARGET AREA.
* A NONZERO RETURN CODE INDICATES AN ERROR.
***********************************************************************
RETURN   DS    0H                     RETURN POINT
         L     R14,12(,R13)           RESTORE R14
         LM    R0,R12,20(R13)         RESTORE REST OF CALLER'S REGS
         BR    R14                    RETURN TO CALLER
         EJECT
PARMLIST DSECT ,                      INPUT PARAMETER LIST
INPTR    DS    F                      POINTER TO INPUT STRING
OUTPTR   DS    F                      POINTER TO TARGET AREA
INLEN    DS    H                      LENGTH OF INPUT STRING
OUTLEN   DS    H                      SIZE OF TARGET AREA
RC       DS    F                      RETURN CODE
         EJECT
DBCSCONV CSECT ,
R0       EQU   00                     EQUATES FOR REGISTERS 0-15
R1       EQU   01
R2       EQU   02
R3       EQU   03
R4       EQU   04
R5       EQU   05
R6       EQU   06
R7       EQU   07
R8       EQU   08
R9       EQU   09
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END   DBCSCONV
```

Figure 112. TRANSPROC Shell (Part 2 of 2)

End of General-Use Programming Interface

Coding Your Own Cancel Exit

When coding your own interactive program to process SQL statements, you may want to code a cancel exit. The ISQL CANCEL command is an example of using a cancel exit. Specifically, it allows you to stop an in-progress command or logical unit of work. When a cancel exit is taken, the database manager stops processing the current SQL statement, and does a ROLLBACK WORK RELEASE for the user who issued the cancel request.

The Online Resource Adapter provides the user cancel exit primarily to allow online applications to perform a CANCEL function. When it would be appropriate to cancel SQL requests (for example, while waiting for an XPCC link or an SQL request), the resource adapter gives control to the user cancel exit. This cancel exit can then do a CICS wait on ECBs pointed to by the RMWL for normal processing to complete or for the terminal operator to request a cancel.

Once posted, the exit can then set the appropriate post-code in the RMARPC field and return to the Online Resource Adapter, which interrogates the post-code and takes the corresponding action.

General-Use Programming Interface

Macro ARIRCAN is a General-Use programming interface. General-Use programming interface is defined in "Programming Interface Information" on page 425.

ARIRCAN Macro

For the convenience of CICS/VSE user transactions to provide a cancel exit, an assembler language macro is provided to generate the RDIIN set exit call. The format of the macro is:

```
  ARIRCAN RMARPTR = addr
```

`where RMARPTR = addr gives the address of a pointer to the RMAR.`

Output from this macro with the RMARPTR operand is an RDIIN, type 165 containing the RMARPTR address in field RDIVPARM. (For more information on the RDIIN, see the *DB2 Server for VSE & VM Diagnosis Guide and Reference* manual.) Also generated is a call to ARIPRDI, which when executed invokes the Online Resource Adapter to perform the Set Exit function.

For the ARIRCAN macro above, the following is required:

1. Invocation must be from a module that has completed assembler preprocessing. This provides for SQLDSECT and SQLCA addressability.
2. As for EXEC SQL, register 13 must point to a standard 72-byte save area, and registers 1, 14, and 15 are modified as a result of the call generated by the macro.

To use the cancel exit for cancel support, an interactive application would need to:

- Define an RMAR as described in the RMAR data area.
- Issue the ARIRCAN macro pointing to the established RMAR. The cancel would be identified in this RMAR through the RMARXP field.

  If your cancel exit is to take advantage of 31-bit addressing, you must set the high order bit of the cancel exit entry point to '1'.

When the online support invokes the cancel exit pointed to by RMARXP, R1 points to a PLIST, the first word of which points to the RMAR. Some actions that the exit might perform are:

- Save registers using standard register conventions. (required)
- Using R1, establish addressability to RMAR.
- Using RMARWLP, establish addressability to RMWL.
- Set RMWLUEP to point to an ECB which is posted by an application cancel routine. When this user written routine recognizes that a cancel request has been made, it posts the ECB pointed to by RMWLUEP. RMWLUEP is immediately followed by pointers to DB2 Server for VSE ECBs, which are to be posted when database activity completes.

- Issue a CICS cancel pointing to RMWLUEP as the beginning of ECB list.
- When the cancel is satisfied, test if the cancel ECB was posted. If so, set RMARPC=8; otherwise, set RMARPC=0. A value of 8 informs the database manager to cancel the current SQL request. A value of 0 informs the database manager to continue processing the SQL request. Any other values in RMARPC results in the SQL request being canceled, SQLERRD1 being set to 4 and SQLERRD2 being set to the value returned in RMARPC.
- Restore the registers and return to the online support via R14 (required).

RMAR (Resource Adapter Asynchronous Request)

This control block is used for EXEC RDIIN set or reset user wait exit requests. An A within the field description indicates that the field is set by the application. An R indicates that the resource adapter sets the field.

For this type of request, the following is applicable in the RDIIN structure.

- RDICTYPE = 165: This is the value reserved for exit calls to the resource adapter.
- RDICALL = “S” for “SET EXIT” and “R” for “RESET EXIT”.
- RDIVPARM = ADDR(RMAR); The RMAR is provided by the application program.
- RDICODEP = ADDR(SQLCA). The following SQLCA values are applicable:
  - SQLCODE=-914 SQLERRD1 = 0 For valid user requested cancel.
  - SQLCODE=-914 SQLERRD1 = 4 For user cancel requested with invalid post code.
  - SQLCODE=-824 For invalid set or reset wait exit requests:
    SQLERRD1=4 When exit already exists.
    SQLERRD1=8 When RDIVPARM=0.
    SQLERRD1=12 When exit pointer is 0.
    SQLERRD1=16 When reset finds no exit to reset.
- RDIERROR should be set to B, E, or W if the application requires the WHENEVER SQL ERROR OR WARNING process to be active for the specific RDIIN exit request. Otherwise, RDIERROR should be set to a blank (“40”X).
- The remainder of the RDIIN structure is not referenced by the resource adapter and should be binary zeros.

RESET has two functions: it allows an application to specify a new exit pointer, and it allows for turning off the exit linkage. For each of these functions, RDICALL=R. If RMARXP is not 0, RMARXP is taken as a new exit pointer. If RMARXP is 0, the resource adapter pointer to the RMAR is nullified.

When an exit has been established, RMLORMAR contains the pointer to RMAR. When the online support invokes an exit, register 1 points to a PLIST, the first work of which points to RMAR.

Dec(HEX) RMAR

0(0)	RMAREYEC - 'RMAR' (eyecatcher)	
8(8)	RMARLEN - RMAR length (A)	Reserved (Binary zeros) (A) (2)
16(10)	RMARAPPL - Non-architected field reserved for the application (A)	RMARXP - application exit
24(19)	RMARPC - Post code (A) (1)	RMARXC - Exit code (R) (2)
32(20)	RMARWLP → RMWL (R)	RMARCNTL - Control Flags (2)
40(28)	RMARUDAT - User data (terminal id)	
48(30)	Reserved (16) bytes	

Figure 113. Online Resource Adapter Asynchronous Request (RMAR)

```
(1) 0: Resource adapter should continue
    4: Reserved
    8: Resource adapter should cancel
    Others: Reserved
(2) See "RMAR flags" that follows.
```

RMAR Flags

```
OFFSET   FIELD NAME   BITS     MEANING

36(24)   RMARCNTL
            RMARWAIT  1... .... Tells ARIRSEND to use the
                                wait exit.
                      0... .... Tells ARISEND to not exit
                                (because call is an EXEC
                                RDIIN PTC or EXEC SQL
                                COMMIT/ROLLBACK WORK (R)
            RMARDSPR  .1.. .... Tells RMAPI to do a
                                ROLLBACK via DFHSP.  Set by
                                ARIRSEN when the attempt
                                to clear did not work
                                because processing already
                                finished.
                      ..xx xxxx Reserved. Binary zeros (R)
```

End of General-Use Programming Interface

Field Procedures

General-Use Programming Interface

A field procedure is a General-Use programming interface. Macro ARIBFPPB is a General-Use programming interface. General-Use programming interface is defined in "Programming Interface Information" on page 425.

A field procedure is a user-written exit routine that transforms values in a single short-string column. When values in the column are changed, or new values are inserted, the field procedure is run to encode each value, which is then stored. When values are retrieved from the column, the field procedure is run to decode each value back to the original string value. A field procedure can be used to alter the sorting sequence of values entered in a column. For example, telephone directories sometimes require that names such as McCabe and MacCabe appear next to each other. This cannot be achieved with the standard EBCDIC sorting sequence. Languages that do not use the Roman alphabet have similar requirements. However, if a column is provided with a suitable field procedure, you can obtain the desired ordering with the ORDER BY clause.

Any indexes defined on a column that uses a field procedure are built with encoded values.

The transformation that a field procedure performs on a value is called *field-encoding*. The same routine is used to undo the transformation when values are retrieved; that operation is called *field-decoding*.

The field procedure is called when a table is created or altered, to define the data type and attributes of an encoded value to the database manager. That operation is called *field-definition*. The data type of the encoded value can be CHAR, VARCHAR, GRAPHIC, or VARGRAPHIC. If the datatype is VARCHAR the length must be 254 or less. If the database is VARGRAPHIC, the length must be 127 or less. For the applicable data types, see the description for the FPVDTYPE field in Table 30 on page 296. The length, precision, or scale of the encoded value must be compatible with its data type. Values in columns with a field procedure are described to the database manager in the following catalog tables:
- SYSTEM.SYSCOLUMNS
- SYSTEM.SYSFIELDS
- SYSTEM.SYSFPARMS
- SYSTEM.SYSKEYCOLS.

For more information about catalog tables, see the *DB2 Server for VSE & VM SQL Reference* manual.

Specifying the Field Procedure

To name a field procedure for a column, use the FIELDPROC clause of the CREATE TABLE or ALTER TABLE statement, followed by the name of the procedure and, optionally, a list of parameters. You can use a field procedure only with a short string column. You cannot add a field procedure to an existing column of a table. You can, however, use the ALTER TABLE statement to add to an existing table a new column that uses a field procedure. (To do so, you would have to unload the data, recreate the table, and load the data back into the table.)

The optional parameter list that follows the procedure name is a list of constants, enclosed in parentheses, called the *literal list*. The literal list is incorporated into a data structure called the *field procedure parameter value list* (FPPVL). That structure is passed to the field procedure during the field-definition operation. At that time, the procedure can modify it or return it unchanged. The output form of the FPPVL is called the *modified FPPVL*; it is stored in the DB2 Server for VSE catalog as part of the field description. The modified FPPVL is passed again to the field procedure when the procedure is called for field-encoding or field-decoding.

When Field Procedures are Called

A field procedure specified for a column is called in three situations:
- For field-definition, when the CREATE TABLE or ALTER TABLE statement that names the procedure is run. When called, the procedure is expected to:
  - Determine whether the data type and attributes of the column are valid

  - Verify the literal list, and change it if required
  - Provide the field description of the column
  - Define the amount of working storage needed by the field-encoding and field-decoding processes.
- For field-encoding, when a column value is to be field-encoded. That occurs for any value that is:
  - Inserted in the column by an SQL INSERT or PUT statement, or loaded by the DBS utility DATALOAD or RELOAD commands
  - Changed by an SQL UPDATE statement
  - Compared to a column with a field procedure, unless the comparison operator is LIKE. The value being encoded is a host variable or constant.
- For field-decoding, when a stored value is to be field-decoded back into its original string value. This occurs for any value that is:
  - Retrieved by an SQL SELECT or FETCH statement, or by the DBS utility DATAUNLOAD or UNLOAD commands
  - Compared to another value with the LIKE comparison operator. The value being decoded is from the column that uses the field procedure.

  In this situation the field procedure is called after any DB2 Server for VSE sort.

A field procedure is never called to process a null value.

General Considerations for Writing Field Procedures

Your field procedure must adhere to the following rules:

- It must be written in Assembler.
- Its name must not start with ARI, to avoid conflict with the DB2 Server for VSE modules.
- It must not call any SVC services.
- It must store registers in an area pointed to by R13, and restore them before returning.
- It must be serially reusable.
- It must not contain any SQL statements.
- It must reside in the appropriate VSE library and be accessible when the database manager is running.
- It must support 31-bit addressing (AMODE 31).

Attention: A field procedure should always transform one input data value into one output data value, unless the parameters are different. This means that the same field procedure with the same parameters must implement a one to one data conversion, in both directions. The field-decoding function must be the exact inverse of the field-encoding function. For example, if a routine encodes ALABAMA to 01, it must decode 01 to ALABAMA. A violation of this rule can lead to unpredictable results and possible data corruption.

A Warning about Blanks

When the database manager compares the values of two strings with different lengths, it temporarily pads the shorter string with blanks (in either single-byte or double-byte characters, as appropriate) up to the length of the longer string. If the shorter string is the value of a column with a field procedure, the padding is done to the encoded value, but the pad character is **not** encoded. Hence, if the procedure changes blanks to some other character, encoded blanks at the end of the longer string are not equal to padded blanks at the end of the shorter string. That situation can lead to errors; for example, some strings that should be equal may not be recognized as such. You should **not** encode blanks with a field procedure.

Maintaining Field Procedures

Field procedures are kept in the appropriate VSE library. They can reside in the PRD2 library as a separate sublibrary. The maximum number of active field procedures on one installation is 16. If this limit is exceeded, an attempt to load a field procedure results in an SQLCODE -682 with reason code 4.

Recovering from Abends in Exits

If a field procedure ends abnormally, a message (ARI0022E) to remove the field procedure from the installation is issued to the operator, the database manager takes a SNAP dump, and processing continues.

Security with Field Procedures

Since exit routines run as extensions of the database manager and have all its privileges, they can impact its security and integrity. All field procedures must be tested and appropriate security measures taken before they are installed on a system.

Field Procedures for Cultural Sorts

By default, string data is sorted based on the S/390 collating sequence. However, the collating sequence required for certain alphabets is different from the default S/390 collating sequence. Users expect that sorted data will match the order that is culturally correct for them and that searches on data will return the result that is correct for the sorting sequence of their language. They are at ease with only one sort order, the one used in their dictionaries, telephone directories, book indices, and so on.

A way to accommodate special sorting requirements is to use Field Procedures. Field Procedures can be used to encode data being inserted into a column. The encoding effectively alters the collating sequence for the data in the column, enabling the special sorting requirements to be met by the S/390 collating sequence.

Two field procedures are provided. The procedures are supplied as A-type members.

The field procedures provided are:
- FP870L2 for Slovenia, Poland and Romania
- FP102CY for Russia, Bulgaria, Serbia and Montenegro

The field procedures are written in Assembler. The field procedure must be assembled and the corresponding phase must be generated and placed in a library that is accessible to the database manager when it is running.

Once the phase for the field procedure has been generated and made accessible to the database manager, it can be used by specifying its name in the FIELDPROC clause of the CREATE TABLE or ALTER TABLE statement.

Field Procedure Interface to the Database Manager

This section describes certain control blocks that are used to communicate to a field procedure, under the following headings:

- "The Field Procedure Parameter List (FPPL)"
- "The Work Area"
- "The Field Procedure Information Block (FPIB)" on page 295
- "Value Descriptors" on page 295
- "The Field Procedure Parameter Value List (FPPVL)" on page 296.

The Field Procedure Parameter List (FPPL)

The FPPL is pointed to by register 1 on entry to a field procedure. It, in turn, contains the addresses of five other areas, shown in Figure 114. The FPPL and the areas to which it points are all described by the mapping macro ARIBFPPB, which is provided as an E-type member.

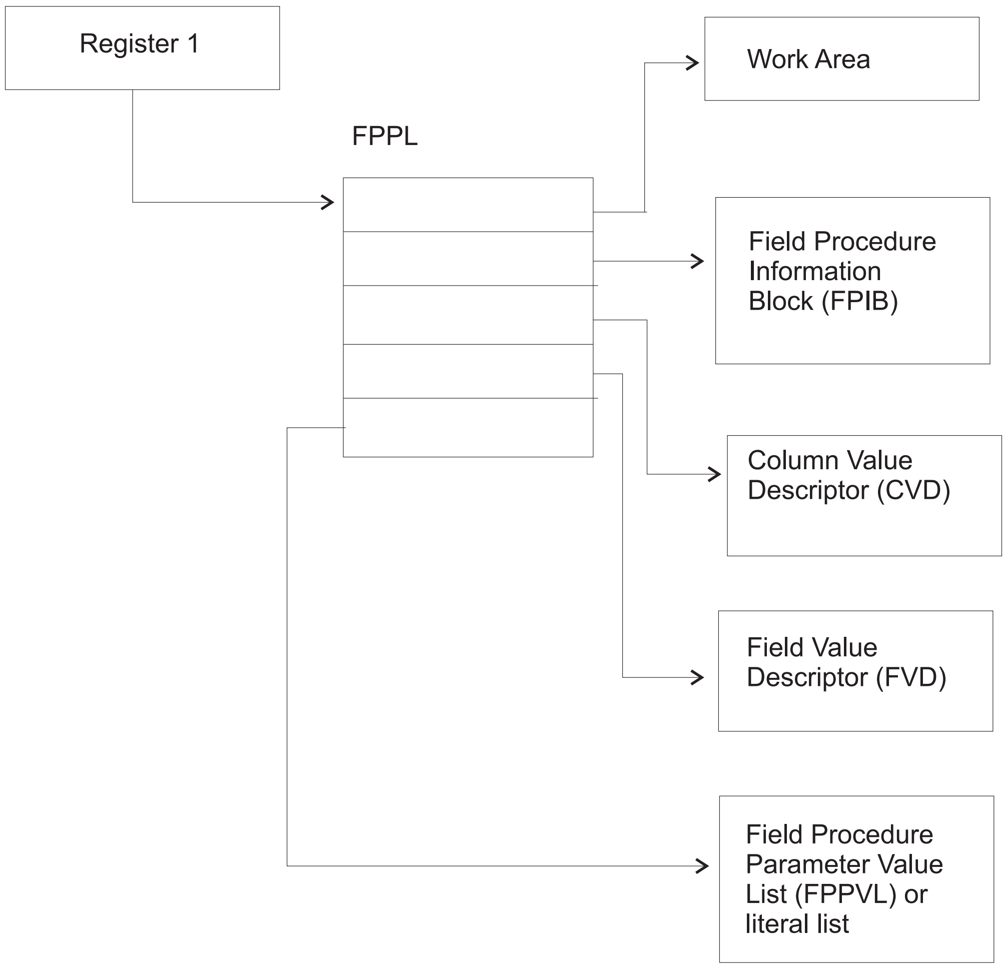

Register 1
FPPL
Work Area
Field Procedure Information Block (FPIB)
Column Value Descriptor (CVD)
Field Value Descriptor (FVD)
Field Procedure Parameter Value List (FPPVL) or literal list

Figure 114. Field Procedure Parameter List

The Work Area

The work area is an area of storage used by a field procedure as working storage. A new area is provided each time the procedure is called.

The size of the area you need depends on the way you have programmed your field-encoding and field-decoding operations. For the field-definition operation, the database manager passes your routine a value of 512 bytes for the length of the work area (FPBWKLN in FPIB). If, for example, the longest work area you need for field-encoding or field-decoding is 1024 bytes, your field-definition operation must change the length to 1024. Thereafter, whenever your field procedure is called for either encoding or decoding, the database manager makes an area of 1024 bytes available to it.

If 512 bytes is sufficient for your operations, your field-definition operation need not change the value supplied by the database manager. If you need less than 512 bytes, your field-definition can return a smaller value. However, your field-definition itself must not use more than 512 bytes.

The Field Procedure Information Block (FPIB)

The FPIB communicates general information to a field procedure. For example, it tells what operation is to be done, allows the field procedure to signal errors, and gives the size of the work area. Its format is shown in Table 29.

Table 29. Format of FPIB, Defined in Copy Macro ARIBFPPB

Name	'Hex' Offset	Data Type	Description
FPBFCODE	0	Signed halfword integer	Function code. **Code** Means **0** Field-encoding **4** Field-decoding **8** Field-definition
FPBWKLN	2	Signed halfword integer	Length of work area; the maximum is 32767 bytes.
	4	Signed halfword integer	Reserved.
FPBRTNC	6	Character, 2 bytes	Return code set by field procedure.
FPBRSNC	8	Character, 4 bytes	Reason code set by field procedure.
FPBTOKP	12	Address	Address of a 40-byte area, within the work area or within the field procedure's static area, containing an error message.

Value Descriptors

Value descriptors describe the data type and other attributes of a value. They are used with field procedures in these ways:

- During field definition, they describe each constant in the field procedure parameter value list (FPPVL). The set of these value descriptors is part of the FPPVL control block.
- During field encoding and field decoding, the decoded (column) value and the encoded (field) value are described by the column value descriptor (CVD) and the field value descriptor (FVD).

The CVD contains a description of a column value and, if appropriate, the value itself. During field encoding, the CVD describes the value to be encoded; during field decoding, it describes the decoded value to be supplied by the field procedure; and during field definition, it describes the column as defined in the CREATE TABLE or ALTER TABLE statement.

The FVD contains a description of a field value and, if appropriate, the value itself. During field-encoding, the FVD describes the encoded value to be supplied by the field procedure; during field-decoding, it describes the value to be decoded. Field-definition must put into the FVD a description of the encoded value.

The format of value descriptors is shown in Table 30.

Table 30. Format of Value Descriptors

Name	'Hex' Offset	Data Type	Description
FPVDTYPE	0	Signed halfword integer	Data type of the value: **Code** Means **16** CHAR **20** VARCHAR **24** GRAPHIC **28** VARGRAPHIC
FPVDVLEN	2	Signed halfword integer	For a varying-length string value, its maximum length.
FPVDVALE	4	None	The value. If the value is a varying-length string, the first half word is the value's actual length in bytes. This field is not present in a CVD, or in an FVD used as input to the field-definition operation.

The Field Procedure Parameter Value List (FPPVL)

The FPPVL communicates the literal list, supplied in the CREATE TABLE or ALTER TABLE statement, to the field procedure during field definition. At that time the field procedure can reformat the FPPVL. The reformatted FPPVL is stored in SYSTEM.SYSFPARMS and communicated to the field procedure during field encoding and field decoding as the modified FPPVL.

Its format is shown in Table 31.

Table 31. Format of FPPVL, Defined in Copy Macro ARIBFPPB

Name	'Hex' Offset	Data Type	Description
FPPVLEN	0	Signed halfword integer	Length in bytes of the area containing FPPVCNT and FPPVVDS. At least 254 for field-definition.
FPPVCNT	2	Signed halfword integer	Number of value descriptors that follow, equal to the number of parameters in the FIELDPROC clause. Zero if no parameters were listed.
FPPVVDS	4	Structure	For each parameter in the FIELDPROC clause, there is: • A signed fullword integer giving the length of the following value descriptor. • A value descriptor.

Field-Definition (Function Code 8)

The input provided to the field-definition operation, and the output required, are as follows:

On ENTRY

The registers have the following information:

Register	Contains
1	Address of the field procedure parameter list (FPPL). For a schematic diagram, see Figure 114 on page 294.
2-12	Unknown values that must be restored on exit.
13	Address of the register save area.
14	Return address.
15	Address of entry point of exit routine.

The contents of all other registers, and of fields not listed below, are unpredictable.

The work area consists of 512 contiguous uninitialized bytes.

The FPIB has the following information:

Field	Contains
FPBFCODE	8, the function code.
FPBWKLN	512, the length of the work area.

The CVD has the following information:

Field	Contains
FPVDTYPE	One of these codes for the data type of the column value:
FPVDVLEN	The length attribute of the column.

Code	Means
16	CHAR
20	VARCHAR
24	GRAPHIC
28	VARGRAPHIC

The FPVDVALE field is omitted.

The FVD provided is 4 bytes long.

The FPPVL has the following information:

Field	Contains
FPPVLEN	The length, in bytes, of the area containing the parameter value list. The minimum value is 254, even if there are no parameters.
FPPVCNT	The number of value descriptors that follow; zero if there are no parameters.
FPPVVDS	A contiguous set of value descriptors, one for each parameter in the parameter value list, each preceded by a 4-byte length field.

On EXIT

The registers must have the following information:

Register	Contains
2-12	The values they contained on entry.
15	The integer zero if the column described in the CVD is valid for the field procedure; otherwise the value must **not** be zero.

Fields listed below must be set as shown; all other fields must remain as on entry.

The FPIB must have the following information:

Field	Contains
FPBWKLN	The length, in bytes, of the work area to be provided to the field-encoding and field-decoding operations; `0` if no work area is required.
FPBRTNC	An optional 2-byte character return code, defined by the field procedure; blanks if no return code is given.
FPBRSNC	An optional 4-byte character reason code, defined by the field procedure; blanks if no reason code is given.
FPBTOKP	Optionally, the address of a 40-byte error message residing in the work area or in the field procedure's static area; zeros if no message is given.

Errors signaled by a field procedure result in an SQL return code of -681, which is set in the SQL communication area (SQLCA). The contents of FPBRTNC and FPBRSNC, and the error message pointed to by FPBTOKP, are also placed into the tokens, in SQLCA, as field SQLERRMT. The meaning of the error message is determined by the field procedure.

The FVD must have the following information:

Field	Contains
FPVDTYPE	The numeric code for the data type of the field value. Any of the data types listed in Table 30 on page 296 is valid.
FPVDVLEN	The length of the field value.

Field FPVDVALE must **not** be set; the length of the FVD is 4 bytes only.

The FPPVL can be redefined to suit the field procedure, and returned as the modified FPPVL, subject to the following restrictions:
- The field procedure must not increase the length of the FPPVL.
- The FPPVLEN must contain the actual length of the modified FPPVL, or `0` if no parameter list is returned.

The modified FPPVL is recorded in the SYSTEM.SYSFPARMS catalog table and is passed again to the field procedure during field-encoding and field-decoding. The modified FPPVL need not have the format of a field procedure parameter list, and it need not describe constants by value descriptors.

Field-Encoding (Function Code 0)

The input provided to the field-encoding operation, and the output required, are as follows:

On ENTRY

The registers have the following information:

Register	Contains
1	Address of the field procedure parameter list (FPPL). For a schematic diagram, see Figure 114 on page 294.
2-12	Unknown values that must be restored on exit.
13	Address of the register save area.
14	Return address.
15	Address of entry point of exit routine.

The contents of all other registers, and of fields not listed below, are unpredictable.

The work area is contiguous, uninitialized, and of the length specified by the field procedure during field-definition.

The FPIB has the following information:

Field	Contains
FPBFCODE	`0`, the function code.
FPBWKLN	The length of the work area.

The CVD has the following information:

Field	Contains
FPVDTYPE	The numeric code for the data type of the column value, as shown in Table 30 on page 296.
FPVDVLEN	The length of the column value.
FPVDVALE	The column value; if the value is a variable-length string, the first halfword contains its length.

The FVD has the following information:

Field	Contains
FPVDTYPE	The numeric code for the data type of the field value.
FPVDVLEN	The length of the field value.
FPVDVALE	An area of unpredictable content that is as long as the field value.

The modified FPPVL produced by the field procedure during field-definition is provided if it exists.

On EXIT

The registers must have the following information:

Register	Contains
2-12	The values they contained on entry.
15	The integer zero if the encoding is successful; otherwise the value must **not** be zero.

The FVD must contain the encoded (field) value in field FPVDVALE. If the value is a varying-length string, the first halfword must contain its length.

The FPIB may have the following information:

Field	Contains
FPBRTNC	An optional 2-byte character return code, defined by the field procedure; blanks if no return code is given.
FPBRSNC	An optional 4-byte character reason code, defined by the field procedure; blanks if no reason code is given.
FPBTOKP	Optionally, the address of a 40-byte error message residing in the work area or in the field procedure's static area; zeros if no message is given.

Errors signaled by a field procedure result in an SQL return code of -681, which is set in the SQL communication area (SQLCA). The contents of FPBRTNC and FPBRSNC, and the error message pointed to by FPBTOKPT, are also placed into the tokens, in SQLCA, as field SQLERRMT. The meaning of the error message is determined by the field procedure.

All other fields must remain as on entry.

Field-Decoding (Function Code 4)

The input provided to the field-decoding operation, and the output required, are as follows:

On ENTRY

The registers have the following information:

Register	Contains
1	Address of the field procedure parameter list (FPPL). For a schematic diagram, see Figure 114 on page 294.
2-12	Unknown values that must be restored on exit.
13	Address of the register save area.
14	Return address.
15	Address of entry point of exit routine.

The contents of all other registers, and of fields not listed below, are unpredictable.

The work area is contiguous, uninitialized, and of the length specified by the field procedure during field-definition.

The FPIB has the following information:

Field	**Contains**
FPBFCODE	4, the function code.
FPBWKLN	The length of the work area.

The CVD has the following information:

Field	**Contains**
FPVDTYPE	The numeric code for the data type of the column value, as shown in Table 30 on page 296.
FPVDVLEN	The length of the column value.
FPVDVALE	An area of unpredictable content that is as long as the column value.

The FVD has the following information:

Field	**Contains**
FPVDTYPE	The numeric code for the data type of the field value.
FPVDVLEN	The length of the field value.
FPVDVALE	The field value; if the value is a varying-length string, the first halfword contains its length.

The modified FPPVL, produced by the field procedure during field-definition, is provided if it exists.

On EXIT

The registers must have the following information:

Register	**Contains**
2-12	The values they contained on entry.
15	The integer zero if the decoding is successful; otherwise the value must **not** be zero.

The CVD must contain the decoded (column) value in field FPVDVALE. If the value is a varying-length string, the first halfword must contain its length.

The FPIB may have the following information:

Field	**Contains**
FPBRTNC	An optional 2-byte character return code, defined by the field procedure; blanks if no return code is given.
FPBRSNC	An optional 4-byte character reason code, defined by the field procedure; blanks if no reason code is given.
FPBTOKP	Optionally, the address of a 40-byte error message

residing in the work area or in the field procedure's static area; zeros if no message is given.

Errors signaled by a field procedure result in an SQL return code of -681, which is set in the SQL communication area (SQLCA). The contents of FPBRTNC and FPBRSNC, and the error message pointed to by FPBTOKP, are also placed into the tokens, in SQLCA, as field SQLERRMT. The meaning of the error message is determined by the field procedure.

All other fields must remain as on entry.

A Sample Exit

Figure 115 on page 303 shows an example of a field procedure.

```
FLCTFLC  TITLE 'DB2 Server for VSE FIELD PROCEDURE EXAMPLE'
FLCTFLC  START 0
FLCTFLC  AMODE 31
FLCTFLC  RMODE ANY
************************************************************
*        DB2 Server for VSE FIELD PROCEDURE TO CONVERT              *
*        FIXED LENGTH CHARACTER TO FIXED                   *
*        LENGTH CHARACTER USING A LOOKUP TABLE             *
************************************************************
         SPACE 3
         PRINT GEN
         USING FLCTFLC,R3              BASE REGISTER
         USING FPIB,R9                 COMMON INFORMATION BLOCK
         USING FPVD,R10                VALUE DESCRIPTOR
         USING FPPL,R11                PARAMETER LIST
         USING WA,R12                  WORK AREA
         USING FPPVL,R8                PARAMETER VALUE LIST
         USING TBLHDRD,R7              TABLE HEADER
         SPACE 3
*************************************************************
*        SET UP MAIN LINE               RETURN R14          *
*************************************************************
         SPACE 3
         SAVE  (14,12),,FLCTFLC
         LR    R3,R15                  LOAD BASE REGISTER
         LR    R11,R1                  PARAMETER LIST POINTER
         L     R12,FPPWORK             WORK AREA ADDRESS
         ST    R13,SAVE13
         L     R9,FPPFPIB              COMMON INFORMATION BLOCK
         MVC   FPBRTNC,=AL2(FPBRC0)    RETURN CODE = 0
         LH    R2,FPBFCODE
         L     R15,FDLFC(R2)           SELECT APPROPRIATE ROUTINE
         LA    R14,RET1
         BR    R15
RET1     DS    0H
         PACK  WADW,FPBRTNC            SET RETURN CODE R15
         CLI   FPBRTNC+L'FPBRTNC-1,C' '
         BNE   NOTBL
         PACK  WADW,FPBRTNC(L'FPBRTNC-1)
NOTBL    DS    0H
         CVB   R15,WADW
         L     R13,SAVE13
         RETURN (14,12),T,RC=(15)
         LTORG
FDLFC    DC    A(ENCODE,DECODE,DEFINE)
         SPACE 3
```

Figure 115. Field Procedure Example (Part 1 of 9)

```
****************************************************************
*        ENCODING ROUTINE                     RETURN R14       *
****************************************************************
         SPACE 3
ENCODE   DS    0H
         MVC   FUNCT,=C'ENCD'
         LA    R5,B1
         B     CHKINP                 CHECK INPUT DESCRIPTION
B1       DS    0H
         LA    R5,B2
         B     CHKOUT                 CHECK OUTPUT DESCRIPTION
B2       DS    0H
         SPACE 3
******************************************************************
*        LOOKUP ROUTINE FOR ENCODING                             *
******************************************************************
         SPACE 3
         L     R10,FPPCVD             INPUT VALUE
         L     R6,TABADDR             TOP OF LOOKUP TABLE
         LA    R5,B3
         B     SETLUP                 SET UP LOOKUP VARIABLES
B3       DS    0H
         SPACE 3
*****************************************************************
*        SET UP LOOP VARIABLES                                  *
*****************************************************************
         SPACE 3
         SR    R4,R4                  CLEAR R4
         IC    R4,ILEN                LENGTH FOR COMPARE
         SH    R4,=H'1'               -1
ITOP     DS    0H
         EX    R4,CLCINST
         BE    IHIT
         A     R6,INCRLEN             INCREMENT TO NEXT ENTRY
         BCT   R13,ITOP
         LA    R13,ER5
         B     ERROR4
IHIT     DS    0H
         L     R10,FPPFVD
         SPACE 3
```

Figure 115. Field Procedure Example (Part 2 of 9)

```
***************************************************************
*       SET UP MOVE INSTRUCTION                               *
***************************************************************
        SPACE 3
        SR    R13,R13                CLEAR R13
        IC    R13,OLEN               OUTPUT LENGTH
        SH    R13,=H'1'              -1
        SR    R5,R5                  CLEAR R5
        IC    R5,ILEN                INPUT LENGTH
        AR    R6,R5                  POINT TO OUTPUT VALUE IN TABLE
        EX    R13,MVCINST
        BR    R14
        SPACE 3
*******************************************************************
*       MOVE AND COMPARE INSTRUCTION FOR EXECUTION INSTRUCTION   *
*******************************************************************
        SPACE 3
        DS    0H
CLCINST  CLC   0(1,R6),FPVDVALE
MVCINST  MVC   FPVDVALE,0(R6)
        SPACE 3
*******************************************************************
*       DECODING ROUTINE                                          *
*******************************************************************
        SPACE 3
DECODE   DS    0H
        MVC   FUNCT,=C'DECD'
        LA    R5,BB1
        B     CHKINP                 CHECK INPUT DESCRIPTION
BB1      DS    0H
        LA    R5,BB2
        B     CHKOUT                 CHECK OUTPUT DESCRIPTION
BB2      DS    0H
        SPACE 3
*******************************************************************
*       LOOKUP ROUTINE FOR DECODING                               *
*******************************************************************
        SPACE 3
        L     R10,FPPFVD             OUTPUT VALUE
        L     R6,TABADDR             TOP OF LOOKUP TABLE
        LA    R5,BB3
        B     SETLUP                 SET LOOKUP VARIABLES
BB3      DS    0H
        SPACE 3
```

Figure 115. Field Procedure Example (Part 3 of 9)

```
*******************************************************************
*        SET UP LOOP VARIABLES                                    *
*******************************************************************
         SPACE 3
         SR    R4,R4                   CLEAR R4
         IC    R4,OLEN                 LENGTH FOR COMPARE
         SH    R4,=H'1'                -1
         SR    R5,R5                   CLEAR R5
         IC    R5,ILEN                 INPUT LENGTH
         AR    R6,R5                   POINT TO OUTPUT VALUE IN TABLE
OTOP     DS    0H
         EX    R4,CLCINST
         BE    OHIT
         A     R6,INCRLEN              POINT TO NEXT ENTRY
         BCT   R13,OTOP
         LA    R13,ER8
         B     ERROR4
OHIT     DS    0H
         L     R10,FPPCVD
         SPACE 3
******************************************************************
*        SET UP MOVE INSTRUCTION                                 *
******************************************************************
         SPACE 3
         SR    R13,R13                 CLEAR R13
         IC    R13,ILEN                INPUT LENGTH
         SR    R6,R13                  POINT TO INPUT VALUE IN TABLE
         SH    R13,=H'1'               -1
         EX    R13,MVCINST
         BR    R14
         SPACE 3
*******************************************************************
*        DEFINE ROUTINE                   RETURN R14              *
*******************************************************************
         SPACE 3
DEFINE   DS    0H
         MVC   FUNCT,=C'DEFN'
         LA    R5,BBB1
         B     CHKINP
BBB1     DS    0H
         SPACE 3
******************************************************************
*        UPDATE WORK AREA LENGTH IN FPIB                         *
******************************************************************
         MVC   FPBWKLN,=Y(WAEND-WA)
         SPACE 3
```

Figure 115. Field Procedure Example (Part 4 of 9)

```
*********************************************************************
*        SET UP FIELD VALUE DESCRIPTOR                              *
*********************************************************************
         SPACE 3
         L     R10,FPPFVD              OUTPUT DESCRIPTOR
         MVC   FPVDTYPE,=Y(FPVDTCHR)   FIXED CHARACTER
         MVI   FPVDVLEN,X'00'
         AH    R10,=H'3'
         MVC   0(1,R10),OLEN
         BR    R14
         SPACE 3
*********************************************************************
*        CHECK INPUT ROUTINE             RETURN R5                  *
*********************************************************************
         SPACE 3
CHKINP   DS    0H
         L     R8,FPPPVL
         L     R10,FPPCVD              INPUT DESCRIPTOR
         CLC   =Y(FPVDTCHR),FPVDTYPE   FIXED CHARACTER ?
         BNE   CHKINPE1
         CLC   FPPVCNT,=H'1'           ONLY ONE PARAMETER ?
         BNE   CHKINPE2                NO, ERROR
         LA    R7,TBLHDR               POINT TO TABLE HEADER TABLE
LOOP1    DS    0H
         CLC   CODE,FPPVVDS+8          IS VALUE IN TABLE
         BNE   CPEND                   NO, INCREMENT
         B     CINCL                   YES, A HIT
CPEND    DS    0H
         AH    R7,=H'8'                EACH TABLE ENTRY 8 BYTES
         CLI   CODE,X'FF'              END OF TABLE?
         BNE   LOOP1                   NO
         LA    R13,ER3
         B     ERROR8                  YES, ERROR
CINCL    DS    0H
         CLC   ILEN,FPVDVLEN+1         CHECK INPUT LENGTH
         BER   R5
         LA    R13,ER4
         B     ERROR4
CHKINPE1 DS    0H
         LA    R13,ER1
         B     ERROR4
CHKINPE2 DS    0H
         LA    R13,ER2
```

Figure 115. Field Procedure Example (Part 5 of 9)

```
ERROR8   DS    0H
         MVC   FPBRTNC,=AL2(FPBRC8)
         B     ERROR
ERROR4   DS    0H
         MVC   FPBRTNC,=AL2(FPBRC4)
ERROR    DS    0H
         MVC   FPBRSNC,FUNCT
         ST    R13,FPBTOKP
         BR    R14
         SPACE 3
*************************************************************
*        CHECK OUTPUT DESCRIPTOR            RETURN R5       *
*************************************************************
         SPACE 3
CHKOUT   DS    0H
         L     R10,FPPFVD              FIELD DESCRIPTOR
         CLC   =Y(FPVDTCHR),FPVDTYPE   FIXED CHARACTER ?
         BNE   CHKOUTE1
         CLC   OLEN,FPVDVLEN+1         CHECK OUTPUT LENGTH
         BER   R5
         LA    R13,ER6
         B     ERROR4
CHKOUTE1 DS    0H
         LA    R13,ER7
         B     ERROR4
         SPACE 3
*******************************************************************
*        SET UP LOOKUP VARIABLE ROUTINE        RETURN R5          *
*******************************************************************
         SPACE 3
SETLUP   DS    0H
         SR    R4,R4                   CLEAR R4
         IC    R4,ILEN                 INPUT LENGTH
         ST    R4,INCRLEN              SAVE INPUT LENGTH
         SR    R4,R4                   CLEAR R4
         IC    R4,OLEN                 OUTPUT LENGTH
         A     R4,INCRLEN              ADD INPUT LENGTH
         ST    R4,INCRLEN              STORE TABLE ENTRY LENGTH
         SR    R13,R13                 CLEAR R13
         IC    R13,NENTR               NUMBER OF ENTRIES
         BR    R5
         SPACE 3
```

Figure 115. Field Procedure Example (Part 6 of 9)

```
*******************************************************************
*        ERROR MESSAGES                                           *
*******************************************************************
ER1      DC    CL40'INVALID COLUMN TYPE'
ER2      DC    CL40'INVALID NUMBER OF PARAMETERS'
ER3      DC    CL40'INVALID PARAMETER VALUE'
ER4      DC    CL40'INVALID COLUMN LENGTH'
ER5      DC    CL40'INVALID INPUT VALUE TO ENCODE'
ER6      DC    CL40'INVALID FIELD LENGTH'
ER7      DC    CL40'INVALID FIELD TYPE'
ER8      DC    CL40'INVALID FIELD VALUE TO DECODE'
         SPACE 3
*********************************************************************
*        TABLE HEADER TABLE                                         *
*********************************************************************
TBLHDR   DS    0F
*********************************************************************
*        FIRST TABLE   CODE = 'A'                                   *
*********************************************************************
         DC    C'A'                    CODE
         DC    X'01'                   INPUT LENGTH
         DC    X'01'                   OUTPUT LENGTH
         DC    X'03'                   NUMBER OF ENTRIES
         DC    A(TABA)                 ADDRESS OF LOOKUP TABLE
*********************************************************************
*        SECOND TABLE   CODE = 'B'                                  *
*********************************************************************
         DC    C'B'                    CODE
         DC    X'04'                   INPUT LENGTH
         DC    X'01'                   OUTPUT LENGTH
         DC    X'22'                   NUMBER OF ENTRIES
         DC    A(TABB)                 ADDRESS OF LOOKUP TABLE
******************************************************************
*        PUT ADDITIONAL TABLE HEADER ENTRIES HERE                *
******************************************************************
         SPACE 3
******************************************************************
*        END OF TABLE HEADERS                                    *
******************************************************************
         DC    X'FF'
         SPACE 3
TABA     DS    0H
         DC    C'H'                    HIGH
         DC    C'7'
         DC    C'M'                    MEDIUM
         DC    C'5'
         DC    C'L'                    LOW
         DC    C'3'
         SPACE 3
```

Figure 115. Field Procedure Example (Part 7 of 9)

```
TABB     DS    0H
         DC    C'AAA '
         DC    X'F0'                   240
         DC    C'AA+ '
         DC    X'E6'                   230
         DC    C'AA  '
         DC    X'DC'                   220
         DC    C'AA- '
         DC    X'D2'                   210
         DC    C'A+  '
         DC    X'C8'                   200
         DC    C'A1  '
         DC    X'BE'                   190
         DC    C'A   '
         DC    X'B4'                   180
         DC    C'A-  '
         DC    X'AA'                   170
         DC    C'BBB+'
         DC    X'A0'                   160
         DC    C'BBB '
         DC    X'96'                   150
         DC    C'BBB-'
         DC    X'8C'                   140
         DC    C'BB+ '
         DC    X'82'                   130
         DC    C'BB  '
         DC    X'78'                   120
         DC    C'BB- '
         DC    X'6E'                   110
         DC    C'B+  '
         DC    X'64'                   100
         DC    C'B   '
         DC    X'5A'                    90
         DC    C'B-  '
         DC    X'50'                    80
         DC    C'CCC '
         DC    X'46'                    70
         DC    C'CC  '
         DC    X'3C'                    60
         DC    C'C   '
         DC    X'32'                    50
         DC    C'D   '
         DC    X'28'                    40
         DC    C'NR  '
         DC    X'1E'
         SPACE 3
```

Figure 115. Field Procedure Example (Part 8 of 9)

```
***********************************************************************
*        TABLE HEADER TABLE DSECT                                     *
***********************************************************************
TBLHDRD  DSECT
CODE     DS    CL1
ILEN     DS    CL1
OLEN     DS    CL1
NENTR    DS    CL1
TABADDR  DS    A
         SPACE 3
**********************************************************************
*        WORK AREA                                                   *
**********************************************************************
         SPACE 3
WA       DSECT
SAVE13   DS    F
INCRLEN  DS    F
FUNCT    DS    CL4
WADW     DS    D
WAEND    DS    0H
         SPACE 3
         ARIBFPPB
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

Figure 115. Field Procedure Example (Part 9 of 9)

End of General-Use Programming Interface

Chapter 14. Using a DRDA Environment

The Distributed Relational Database Architecture (DRDA) environment provides the architecture for access to data that is distributed across different operating systems. The application requester and the application server do not have to be running with the same database manager.

This chapter discusses:
- Benefits and added responsibilities of a DRDA environment
- Types of distributed access
- Preparing to implement DRDA
- Installing and removing the DRDA code
- Using DRDA
- Creating the DBS Utility on remote DRDA application servers
- Using ISQL on remote DRDA application servers
- Two phase commit processing

Not all extended features are supported by the DRDA protocol. Refer to Appendix H, "DRDA Considerations," on page 385 for more details.

For detailed information on Distributed Relational Database Architecture, see the manuals in the *Distributed Relational Database Architecture Library* listed in the Bibliography.

Benefits of Using the DRDA Protocol

The DRDA option does the following:
- Makes DB2 Server for VSE data accessible to users equipped with the DRDA application requester function. Users on platforms such as OS/2, AIX, OS/400, OS/390, or Microsoft Windows™ can run applications that utilize DRDA remote unit of work or DRDA distributed unit of work processing to access data residing in DB2 Server for VSE application servers.
- Enables DB2 Server for VSE users to use remote unit of work access to work with data on non-DB2 Server for VSE application servers. This allows access to data that would otherwise remain unavailable.

To support this access, application programs can contain SQL statements that are specific to the target system, and both the DBS Utility and ISQL can be run on non-DB2 Server for VSE application servers. The SQL statements in these application programs can be static, dynamic, and extended dynamic, even if the target system does not support extended dynamic statements. In addition, portable packages can be loaded on non-DB2 Server for VSE application servers.

The DRDA option provides the following additional functions:
- To determine the status of connections in an environment that may have local and remote systems, you can use the SHOW CONNECT operator command.
- To aid in the diagnosis of errors, first failure data capture is automatically performed. IBM service can use the captured data for diagnosis, decreasing the probability of having to rerun applications to acquire data for diagnosis.
- Another aid in the diagnosis of errors is the LUWID support. The LUWID is a unique identifier associated with each application requester connection. It is

composed of four parts: network id, LU name, LUW instance number, and LUW sequence number. This provides additional information that may be required in problem diagnosis.

Added Responsibilities in Using the DRDA Protocol

Use of the DRDA protocol requires assuming extra responsibilities that are usually not required in a non-distributed environment.

Because the communications between database managers can be in different time zones or countries, some allowance must be made for scheduling and communication problems (particularly when different languages are involved).

The operation of applications may be similar, but the different platforms will require modifications. These modifications may require that as system administrator you become familiar with the terminology used on non-DB2 Server for VSE database managers. In situations such as adding users, assigning resources, ascertaining the authorization schemes available, and performing diagnosis, the different terminology of the different database managers can lead to misunderstandings. Similarly, because communications software is involved, you may have to become familiar with communication terminology that may not be required in a non-distributed environment.

Applications that run in a DRDA environment also require attention. In some instances, they may have to be recoded to compensate for system-to-system processing differences. As an example, consider the differences between collating sequences on different database managers. Quite apart from the differences between the ASCII and EBCDIC collating sequences, differences can occur between EBCDIC collating sequences on two different database managers: the same character can appear in a different sequence because of the way in which a system processes information. If an application is not recoded to correct for this variability, the results generated by that application can be misleading.

Types of Distributed Access

Two types of access to data in distributed relational database systems are currently available. They are *remote unit of work*, which is also known as DRDA1, and *distributed unit of work*, which is also known as DRDA2.

Remote Unit of Work

Remote unit of work (RUOW) allows a user or an application to read or update data at one remote location per unit of work. With remote unit of work, you can have many SQL statements within a unit of work. You can access one database management system with each SQL statement, and you can access one database management system within a unit of work.

Consider a banking example. Using remote unit of work, you can transfer funds from a savings account table to a checking account table, if both tables are at the same remote location. Figure 116 on page 315 shows how the application first requests an update to the savings account table (1) and then requests an update to the checking account table (2).

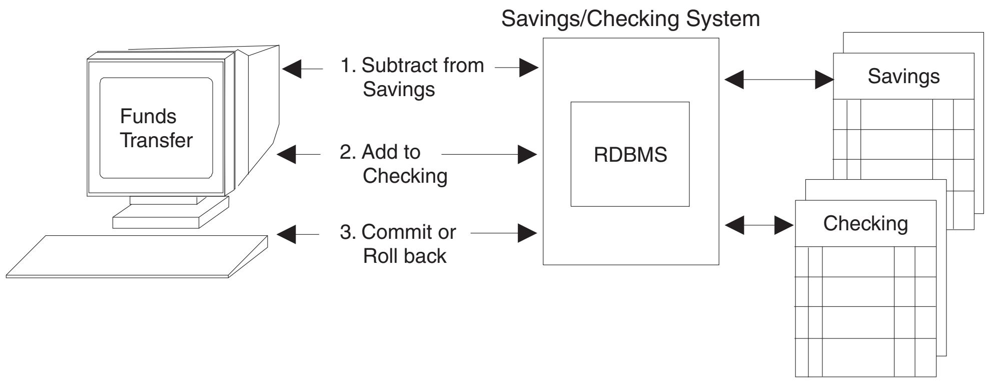


Figure 116. Remote unit of work

If both requests are processed successfully, the application can direct the database management system to commit both updates (3). If either request is not processed successfully, the application can issue a ROLLBACK, leaving both tables as they were before the transaction began. This ensures that requests are neither lost nor duplicated.

Distributed Unit of Work

Distributed unit of work lets a user or application program read or update data at multiple locations within a single unit of work. With distributed unit of work, you can:

- Have many SQL statements within a unit of work
- Access one database management system with each SQL statement
- Access many database management systems within a unit of work.

Using the banking example (see Figure 117), imagine that the savings account table and the checking account table are on two different computer systems. Distributed unit of work processing permits an application to debit the savings account (1), credit the checking account (2), and either commit or roll back the operations in both computer systems (3), treating all of the changes as a single transaction, or unit of work.

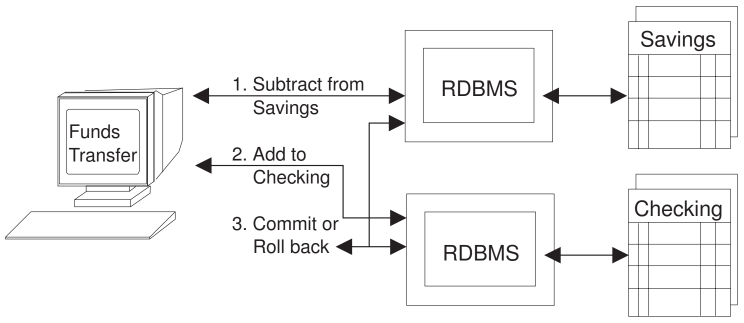


Figure 117. Distributed unit of work

Commit and rollback are coordinated at all locations so that if a failure occurs anywhere in the system, data integrity is preserved. If there was a failure in the middle of the banking transaction just described, and commit or rollback was not coordinated, the savings account could be debited money and the checking account might not be credited the money. This costly error is avoided by the coordination of commit and rollback, or *two-phase commit processing*. For more information on two-phase commit processing, see “Two-Phase Commit Processing” on page 325.

Summary of DRDA Support in DB2 Server for VSE

Table 32 summarizes the level of DRDA support available for the DB2 Server for VSE application server (AS) and application requester (AR):

Table 32. DRDA Support in DB2 Server for VSE

	VM or VSE AS	VM AR	VSE Batch AR	VSE Online AR
RUOW over SNA	yes	yes	no	yes
RUOW over TCP/IP	yes	yes	yes	yes
DUOW over SNA	yes	no	no	no
DUOW over TCP/IP	no	no	no	no

Preparing to Implement DRDA

You can use the application requester, the application server, or both in a distributed environment. This section provides a checklist of the required tasks for implementing DRDA over SNA. For information on implementing DRDA over TCP/IP, refer to Chapter 15, “Using TCP/IP with DB2 Server for VSE,” on page 335. For detailed information on DRDA, see the *Distributed Relational Database Connectivity Guide*.

On the Application Requester

The following tasks must be completed before a batch or online application requester can access a remote server via TCP/IP:

- The DRDA code must be activated. See “Installing the DRDA Code on the Application Requester” on page 320 for more details.
- VSE TCP/IP support must be installed and enabled.
- The DBNAME Directory must be updated to identify the remote application server as accessible via TCP/IP.
- Optionally, the SQLGLOB file can be updated with default parameters for application requesters accessing remote servers.

The following tasks must be completed before a CICS online application requester can access a remote DRDA application server via SNA:

- Update the DBNAME Directory with the remote application server’s SNA information
- Issue CEDA DEFINE CONNECTION (or equivalent DFHCSDUP) to define the remote LU associated with the remote application server. See Figure 12 on page 31 for an example of the CEDA DEFINE CONNECTION.
- Issue CEDA DEFINE SESSION to define the LU 6.2 sessions with the remote system.
- Define the CCSIDs-related phases to CICS. Figure 118 on page 317 shows the DFHCSDUP commands for the new CCSIDs-related programs that must be added.

```
* Phase for SYSCCSIDS
DEFINE PROGRAM(ARISCCSD) GROUP(DB2710) LANGUAGE(ASSEMBLER)
* Phase for SYSSTRINGS
DEFINE PROGRAM(ARISSTRD) GROUP(DB2710) LANGUAGE(ASSEMBLER)
* Phase for SYSCHARSETS
DEFINE PROGRAM(ARISSCRD) GROUP(DB2710) LANGUAGE(ASSEMBLER)
```

Figure 118. Sample commands of the DFHCSDUP command to define a program

- Define the Online Resource Adapter DRDA Router program ARI0RTRM to CICS. Figure 119 shows the DFHCSDUP commands for the new program that must be added.

```
* Phase for Online Resource Adapter DRDA Router
DEFINE PROGRAM(ARI0RTRM) GROUP(DB2710) LANGUAGE(ASSEMBLER)
```

Figure 119. Sample commands of the DFHCSDUP command to define a program

- Activate the DRDA code for the Online Resource Adapter. See “Installing the DRDA Code on the Application Requester” on page 320 for more information.
- Issue CIRB or CIRA to enable online access to the remote application server.
- Issue DSQU (if a CICS user) to override any default global SQLGLOB parameter settings, if required. The SQLGLOB environmental parameters are used by the Online Resource Adapter when accessing a remote application server. Note: When the Online Resource Adapter processes an SQL request from ISQL, CBND, or any task started by the EXEC CICS START command, the Online Resource Adapter uses the default global SQLGLOB parameter values, instead of the user SQLGLOB parameter values.

On the Application Server

Several CICS definitions are required in order to use DRDA support on the VSE application server. See the following sections for more details.

CICS Transaction Definitions Required for DRDA

All DB2 Server for VSE transactions that are used for DRDA processing must be defined to CICS using the Resource Definition Online (RDO) facility or DFHCSDUP commands. Each Transaction Program Name (TPN) in the DBNAME directory that is an APPC-to-XPPC exchange (AXE) transaction must be defined to CICS. The default DBNAME directory entry defines a TPN with a hexadecimal value of X’07F6C4C2’. To define this default TPN to CICS, refer to the CAXE entry in Table 33 on page 318. Additional AXE transactions can be added with different TRANS IDs and, optionally, the TCLASS parameter to provide access control. The TCLASS parameter is not required for the default CAXE entry. To define an alphanumeric TPN, refer to the CAX2 entry in Table 33 on page 318.

The DB2 Server for VSE DRDA2 administration routines must also be defined to CICS if you intend to establish protected conversations between a remote client and the DB2 Server for VSE application server. This includes the DB2 Server for VSE DAXP and DAXT transactions. The DAXP transaction sets parameters that are used when an AXE transaction subsequently autostarts DRDA2 support. The DAXT transaction is responsible for disabling DRDA2 support.

Use the sample entries in Table 33 on page 318 as a guideline for making your transaction definitions.

Table 33. Defining the DB2 Server for VSE DRDA Transactions

TRansaction	PROGram	TWasize	Xtranid	SPurge	TPurge
DRDA Server Support AXE Transaction					
CAXE	ARICAXED	0	07F6C4C2	YES	YES
CAX2	ARICAXED	0		YES	YES
DRDA2 Parm Setting Entry					
DAXP	ARICDAXD	0		YES	YES
DRDA2 Disable TRUE Entry					
DAXT	ARICDAXD	0		YES	YES

CICS Programs Required for DRDA

DB2 Server for VSE programs that support DRDA processing and run in the CICS partition must be link-edited and defined to CICS using the Resource Definition Online (RDO) facility or DFHCSDUP commands.

The following programs must be link-edited and defined to CICS, if you have remote clients accessing the DB2 Server for VSE application server:

- The AXE transaction
- AXE TRUE support, to obtain the LUWID for the AXE transaction
- DBName Directory services

In addition, if you intend to establish a protected conversation between a remote client and the DB2 Server for VSE application server, you must link-edit and define the following programs to CICS:

- DRDA2 administration routines (DAXP and DAXT) responsible for updating the DAXP parameter table and for disabling the DRDA2 TRUE support
- DRDA2 TRUE support enabling routine
- DRDA2 TRUE support
- DRDA2 DAXP parameter control block (DR2DFLT)

Use the sample entries in Table 34 to make your definitions to CICS. The column ″Link Book″ specifies which DB2 Server for VSE link book to use when link-editing a specific program.

Table 34. Defining the DB2 Server for VSE DRDA Programs

Program	Description	Link Book	Resident	Language
ARICAXED	APPC-XPCC Exchange (AXE)	ARISLK9D	YES	ASSEMBLER
ARICAXLD	AXE TRUE Support	ARISLK9Z	YES	ASSEMBLER
ARICDIRD	DBNAME Directory Services	ARISLKDA		ASSEMBLER
ARICDAXD	DAXP and DAXT	ARISLKND	YES	ASSEMBLER
ARICDEBD	DRDA2 TRUE Enabling	ARISLKVD		ASSEMBLER
ARICDRAD	DRDA2 TRUE Support	ARISLKOD		ASSEMBLER
ARICDR2	DR2DFLT Control Block	ARISLKWD		ASSEMBLER

Entries Required in DFHSIT

In order for CICS/VSE to be accessible to the SNA network as an LU (logical unit of type 6.2), you must define the name of the LU using the APPLID parameter of the DFHSIT macro. The name specified must be the same as the name specified on the VTAM "APPL" definition when CICS was defined to VTAM. Also, if you are using user-defined AXE entries in the DFHPCT table and specifying the TCLASS parameter, the DFHSIT macro must include the CMXT parameter to provide access control.

Terminal Definitions Required by AXE

Each remote DRDA system must be defined to CICS as remote LU 6.2 terminals by updating the CICS System Definitions as follows:

- The remote system itself is defined with the DEFINE CONNECT definition.
- Use DEFINE SESSIONS to define the session characteristics for the remote systems.

Refer to the *CICS/VSE Intercommunication Guide*, and the *CICS/VSE Resource Definition (Online)* for complete information on defining remote systems in CICS.

Entries Required in DFHSNT

Every user ID and password used by a remote DRDA requester must be defined to CICS in the DFHSNT table. Using the DFHSNT TYPE=ENTRY macro, define the user ID using the USERID parameter and define the password using the PASSWRD parameter.

CICS Transaction Server (TS) Considerations

CICS internal security and therefore the CICS Sign-On table (DFHSNT) has been withdrawn in CICS TS for VSE/ESA 2.4. Instead, any external security manager (ESM) may be used that conforms to the VSE/ESA RACROUTE interface. Alternatively, the basic form of external security manager (BSM) provided with VSE/ESA 2.4 may be used. The BSM provides sign-on and transaction attach security only.

When using an ESM, refer to the relevant documentation supplied with the ESM on defining DB2 for VSE user id(s) and operator data and transaction security.

If the BSM is used, DB2 for VSE userids in the DFHSNT and DB2 for VSE transactions must be defined using the VSE Interactive Interface (II).

As the CICS TS System Definition (CSD) file is not compatible with earlier versions of CICS (for example, CICS 2.3), a separate CSD file must be defined and all DB2 for VSE entries added using the DFHCSDUP utility provided with CICS TS.

For DB2 for VSE with CICS TS, the following System Initialization (DFHSIT) parameters are obsolete.

```
AMXT=
EXEC=  (command level is mandatory)
EXITS= (the user-exit interface is always enabled)
MONITOR= (replaced by new monitoring parameters)
```

Refer to the *CICS/VSE Release Guide* manual for a complete list of obsolete DFHSIT parameters.

For more information on installing DB2 for VSE V6.1 with CICS TS, refer to the following publications:

- *CICS/VSE Release Guide, GC33-1645*
- *CICS/VSE Migration Guide, GC33-1646*
- *CICS/VSE Resource Definition Guide, SC33-1653*

Installing and Removing the DRDA Code

Installing the DRDA code is an optional customization step that follows either installation or migration. You install it:

- Immediately after installing or migrating the base code
- At a later date, whenever it is required
- On either the application server or the application requester, or both

You can remove the DRDA code if it is no longer required.

When the DRDA code is installed on the application server, access from DB2 Server for VSE and non-DB2 Server for VSE application requesters is allowed.

When the DRDA code is installed on the application requester, access to remote application servers is allowed.

Do not install the DRDA code unless it is specifically required, as the additional code required for distributed communications requires a significant amount of storage. For details on virtual storage requirements, see Appendix A, “Processor Storage Requirements,” on page 339.

Installing the DRDA Code on the Application Server

To enable DRDA server support, execute job control member ARIS732D. This job can be executed any time when DRDA server support is to be enabled. The support is activated the next time the DB2 Server for VSE database manager is started.

Removing the DRDA Code on the Application Server

To disable DRDA server support, the job control member ARIS733D can be executed any time. The support is deactivated the next time the DB2 Server for VSE application server is started.

Installing the DRDA Code on the Application Requester

There are two linkbooks for the Online Resource Adapter. The first linkbook is used when the shipped Online Resource Adapter phase is linkedited. The second linkbook is used to complete the enablement of the DB2 Server for VSE online DRDA application requester support.

To enable DRDA online application requester support, execute job control member ARIS735D. This job can be executed any time when DRDA online application requester support is to be enabled. The support is activated the next time the Online Resource Adapter is recycled.

To enable DRDA batch application requester support, execute job control member ARIS73LD.

Removing the DRDA Code on the Application Requester

To disable DRDA online application requester support, execute job control member ARIS736D. This job can be executed at any time when DRDA online application requester support is to be disabled. The support is deactivated the next time the Online Resource Adapter is recycled.

To disable DRDA batch application requester support, execute job control member ARIS73MD.

Using DRDA

For the DRDA code to be used on the VSE application server the following conditions must be met:

- The application server DRDA code must be linkedited. See "Installing the DRDA Code on the Application Server" on page 320 for more information.
- A non-zero value must be specified for the database startup parameter RMTUSERS. The RMTUSERS parameter specifies how many application requesters can connect to the VSE application server concurrently.

The recommended supervisor modes for operation are the ESA mode or the VMESA mode. DRDA support is not provided with the VSE Guest Sharing function.

For the DRDA code to be used on the VSE application requester the following conditions must be met:

- The application requester DRDA code must be linkedited. See "Installing the DRDA Code on the Application Requester" on page 320 for more information.
- The DBNAME Directory entry for the database that the VSE application requester will access must indicate that the database is remote. The VSE application requester always uses DRDA protocol to access remote databases. The DBNAME Directory entry contains other important information as well. For more information, see "Setting Up the DBNAME Directory" on page 23.

CICS/VSE online application programs and VSE batch application programs have the ability to execute SQL statements to access and manipulate data managed by any remote application server that implements the DRDA architecture. The SQL statements in these application programs can be static, dynamic, and extended dynamic, even if the target system does not support extended dynamic statements.

Note: Application programs accessing a local AS (or a VM database via guest sharing) will always use Private protocol.

Application programs use the facilities of an accessible DB2 Server for VSE Online Resource Adapter, running in the same partition as the application and acting as a DRDA application requester, to route SQL requests to a DRDA application server. This is illustrated in Figure 120 on page 322.

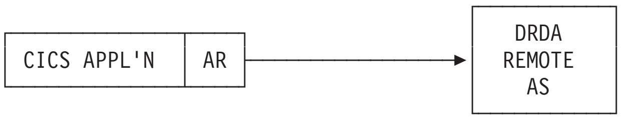


Figure 120. Online DRDA Application Requester (AR) Support

The batch application programs use the facilities of the Batch Resource Adapter which executes in the same partition as the batch application program. The batch application requester is loaded into the partition when the first SQL request is issued by the application program.

The Online Resource Adapter establishes communication links to local application servers at initialization time and maintains these links. CICS applications accessing the local application servers use these links. For remote application servers, the Online Resource Adapter does not establish any communication links at initialization time. Instead, the Online Resource Adapter acquires a session to the remote system where the remote server runs when the application program first connects to the remote server. The session is freed when either:
- the application program ends, or
- the application program switches to another server, or
- the application program switches to another authorization id

The Batch Resource Adapter establishes communication links to local or remote application servers as needed. These links are freed from either:
- the application program ends, or
- the application program switches to another server, or
- the application program switches to another authorization id

An application program can access only one application server (remote or local) in a single unit of work (LUW). A COMMIT RELEASE or ROLLBACK RELEASE must be issued to terminate the LUW before an attempt is made to connect to another application server.

In addition to the remote server it is updating, a CICS/VSE application program can, within the same unit of work, update another CICS resource which participates in two-phase commit processing. Note that VSAM does not participate in two-phase commit processing. In this case, the user executing the program must specify a value of 2 for the SQLGLOB parameter SYNCPOINT. The CICS/VSE syncpoint manager establishes a protected conversation with the remote server and the CICS/VSE syncpoint manager ensures that updates made to the remote server and these other CICS resources are synchronized. Note that to the remote DRDA application server this connection looks like a DRDA 2 connection. However the CICS/VSE application is still limited to accessing a single DRDA server within one LUW. That is, the CICS/VSE application is not able to use CONNECT (Type 2) connections as defined in the *IBM SQL Reference, Version 2, Volume 1*.

Batch applications always use single phase commits (SYNCPOINT 1).

Creating Packages on the Remote Server

If an application program is to access a remote application server, a package corresponding to the application program must be created in the remote application server. This can be done in one of two ways:

- Preprocess the program directly against the remote application server by using the DBNAME preprocessing parameter
- Preprocess the program to create a bind file, then use the on-line transaction CBND or Batch Binding to bind the package to the remote application server.

The DB2 Server for VSE preprocessor can create a package in a single remote application server. Also, the DB2 Server for VSE preprocessor can generate an optional bind file, in addition to the package it creates on the application server. The bind file contains the preprocessor options and the SQL statements from the application program. This information is used by the online binder CBND and Batch Binder to create a package in a remote (or local) application server in the online and batch environment respectively. For more information on how to create a bind file and use the online bind utility (CBND) and Batch Binding, see the *DB2 Server for VSE & VM Application Programming* manual.

Using the DBS Utility on Remote Application Servers

For a user to be able to use the DBS utility on a remote DRDA target application server, you must first preprocess the DBS utility package ARIDSQLP or create the DBSU package using VM Binding facility on the target application server and then create the table SQLDBA.DBSOPTIONS on that application server. This is done by the DB2 Server for VSE application requester. You must then obtain the necessary program bind and table creation privileges for your authorization-id on the target application server.

Note: If the target application server does not support the ERROR option when preprocessing, you must create the DB2 Server for VSE & VM system catalog tables on the target application server for the preprocessing to work. The database managers that do not support the ERROR option (such as the common server database managers) generally supply a command file that creates the necessary table definitions. The command file to create the tables for the DBS Utility is typically called SQLDBSU.CMD or SQLDBSU.BAT.

To create the DBS Utility package, do the following from a DB2 Server for VSE application requester:

1. Ensure that the remote server is identified in the application requester's DBNAME Directory and can be accessed via a TCP/IP network.
2. Preprocess the DBS Utility against the remote application server to create the DBS Utility package. Use the preprocessor options 'PREP=SQLDBA.ARIDSQL,BLK,ISOL(CS),NOPR,NOPU,CTOKEN(NO),ERROR' (omit the ERROR option if the target application server does not support it). Use the member 'ARIDSQLP.A' as the input to the preprocessor. See the *DB2 Server for VSE & VM Application Programming* manual for more information on preprocessing.
3. If you ran a command file to create the table definitions necessary for preprocessing, the DBSOPTIONS table should have been created for you. If this table does not exists, enter the following DBS Utility commands:

```
SET ERRORMODE CONTINUE;

CREATE TABLE SQLDBA.DBSOPTIONS
 (SQLOPTION VARCHAR (18) NOT NULL,
  VALUE     VARCHAR (18) NOT NULL);

CREATE UNIQUE INDEX SQLDBA.DBSINDEX
 ON SQLDBA.DBSOPTIONS (SQLOPTION,VALUE);
```

```
     INSERT INTO SQLDBA.DBSOPTIONS
      VALUES ('RELEASE','7.1.0');

     COMMIT WORK;
```

You must now obtain the necessary program bind and table creation privileges for your authorization-id on the target application server.

To create the DBSU package using VM Binding facility, do the following:

1. Store the DBSU bind file, which was initially shipped as an 80-byte A-type source member (ARIDSQLB), in the ″DB2.BIND.MASTER″ bind file or in your private VSAM file.

   For more information on storing the DBSUL bind file in the ″DB2.BIND.MASTER″ bind file, see *DB2 Server for VSE Program Directory*.
2. Binding can be done using online transaction CBND or Batch Binding.
3. For online binding, execute CBND, specifying the package name SQLDBA.ARIDSQL, against the target application server.
4. Batch binding can be done by invoking ARIPBIN phase through a jcl and specifying the package name and bindfile.

Create the DBSOPTIONS using DBSU. Enter the following DBS Utility commands:

```
SET ERRORMODE CONTINUE;
CREATE TABLE SQLDBA.DBSOPTIONS
(SQLOPTION VARCHAR (18) NOT NULL,
VALUE VARCHAR (18) NOT NULL);
CREATE UNIQUE INDEXSQLDBA.DBSINDEX
ON SQLDBA.DBSOPTIONS (SQLOPTION,VALUE);
INSERT INTO SQLDBA.DBSOPTIONS
VALUES ('RELEASE','7.5.0');
COMMIT WORK;
```

Using ISQL on non-DB2 Server for VSE Application Servers

For a user to be able to make ISQL requests against a remote application server, you must create the ISQL package on the remote application server. You can use the DBS Utility RELOAD PACKAGE command or online/batch binding facility to do this.

Job ARIS120D can be used to create the ISQL package using the DBS Utility.

To create the ISQL package using binding, do the following:

- Store the ISQL bind file, which was initially shipped as an 80-byte A-type source member (ARISIQBD), in the ″DB2.BIND.MASTER″ bind file or in your private VSAM file. . For more information on storing the ISQL bind file in the ″DB2.BIND.MASTER″ bind file, see the *DB2 Server for VSE Program Directory*.
- Binding can be done using online transaction CBND or Batch Binding.
- For on-line binding, execute CBND, specifying the package name SQLDBA.ARIISQL, against the target application server. For more information on the CBND transaction and Batch Binding, see the *DB2 Server for VSE & VM Application Programming* manual. If the target application server does not support the ERROR option, it should supply a command file named ISQL.CMD or ISQL.BAT, which creates the tables that must exist in order to create the ISQL package.
- Batch binding can be done by invoking ARIPBIN phase through a jcl and specifying the package name and bindfile.

- Create the table SQLDBA.ROUTINE, and any other userid.ROUTINE tables that you want.

Two-Phase Commit Processing

Distributed unit of work is a coordinated approach involving two phases. This coordination is done by a *sync point manager*. DB2 Server for VSE uses CICS/VSE as its sync point manager. A sync point manager maintains consistency in changes which are made to protected resources. The primary functions of a sync point manager include, but are not limited to, the following:

1. Keeping track of and logging LUW state information
2. Keeping track of and logging all local protected resource manager (PRM) names that are involved with a logical unit of work
3. Coordinating the COMMIT and ROLLBACK of all local PRMs
4. Initiating resynchronization protocols for any logical unit of work that may be in the in-doubt state because of a system or communications failure.

A sync point manager is required wherever resources may be updated. This requires that sync point managers at each distributed location communicate with one another using architected protocols. These protocols are fully discussed in the *SNA LU 6.2 Reference: Peer Protocols* manual.

For a full explanation of what two-phase commit is, see the following manuals:

- *IBM Systems Network Architecture, Format and Protocol Reference*
- *Reference Manual: Architecture Logic for LU Type 6.2*
- *IBM Systems Network Architecture, Logical Unit 6.2 Reference: Peer Protocols*
- *IBM Distributed Relational Database Architecture Reference*
- *Distributed Data Management (DDM) General Information*.

Using the Two-Phase Commit Protocol

An example of a two-phase commit protocol sequence is shown in Figure 121 on page 326. SNA LU 6.2 functions provide so many capabilities that it is impossible to show all the possible sequences. Notes describing key points in the sequence follow the sequence diagram.

The following assumptions have been made for the example:

- A conversation has been successfully established between the Source Server and the target communications manager (TCM) using a protected conversation.
- No error situation occurs.

For example:

- The ″Source Server″ could be DDCS Multi-User Gateway V2.3.1. In this case, the ″SYNCPNTMGR″ would be function included with DDCS. Also, the ″SNA LU 6.2″ function could be provided by Communications Server for OS/2 Version 4.
- The ″Target Server″ would be DB2 Server for VSE. The ″TCM″ is the AXE transaction. The ″SYNCPNTMGR″ would be CICS/VSE. The ″Other Protected Managers″ would be the database manager function of DB2 Server for VSE.

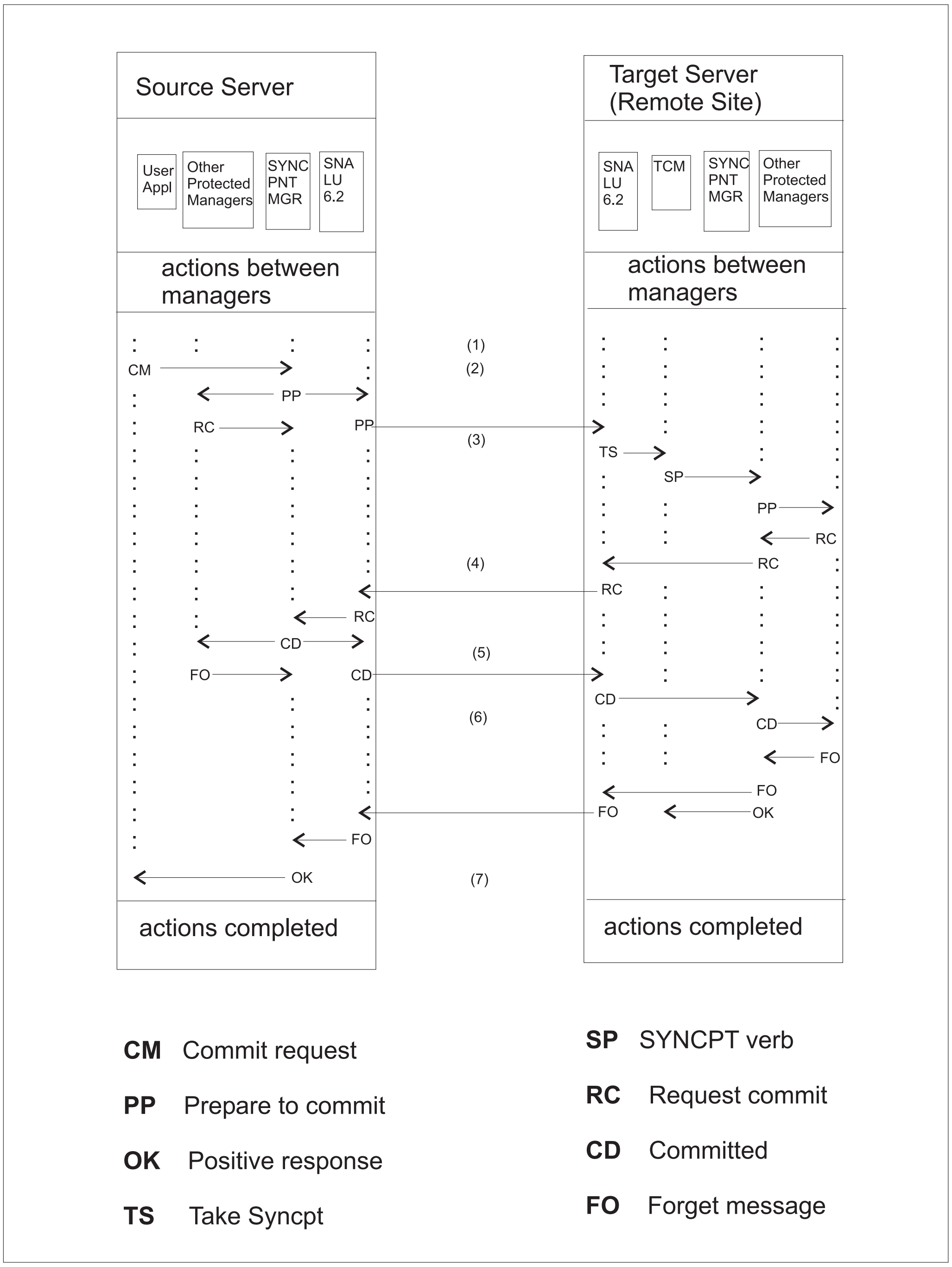


Figure 121. Successful Two-Phase Commit

Figure Notes:

(1) The Target Communications Manager (TCM) issues a RECEIVE_AND_WAIT APPC verb to receive the next SQL Request from the Source Server.

(2) The source application program requests the SYNCPNTMGR to commit the logical unit of work (LUW). The source SYNCPNTMGR notifies the SNA LU 6.2 communications facilities to prepare to commit and notifies the source database (and other protected resource managers registered with the SYNCPNTMGR) to prepare to commit. The source communications facility sends the SNA LU 6.2 prepare message to the target system. The local protected resource managers respond to the source SYNCPNTMGR with the ″Request Commit″ message.

(3) On the target system, the RECEIVE_AND_WAIT verb is completed and the WHAT_RECEIVED parameter is set to TAKE_SYNCPT.

The TCM issues a SYNCPT verb to the target SYNCPNTMGR which begins the commit processing. The SYNCPNTMGR prepares the protected resources to commit.

(4) The SYNCPNTMGR sends the SNA LU 6.2 request commit message to the source system.

(5) The source SYNCPNTMGR collects the request commit messages from the SNA LU 6.2 communications facilities and the other protected resource managers. The source SYNCPNTMGR then commits the logical unit of work by requesting that all of the resources commit. This causes an SNA LU 6.2 committed message to be sent to the target system.

(6) The target SYNCPNTMGR requests that the local resources commit the logical unit of work and causes an SNA LU 6.2 forget message to be sent to the source system. In addition, the target SYNCPNTMGR posts a positive response to the TCM for the SYNCPT verb issued in note (3).

(7) When the source SYNCPNTMGR receives the ″FO″ responses from the protected resource managers, a positive response to the commit is given to the application program.

CICS/VSE Syncpoint Manager and the Task Related User Exit (TRUE)

In the DRDA2 VSE environment, DB2 Server for VSE uses CICS/VSE as its sync point manager. The environment can be shown as follows:

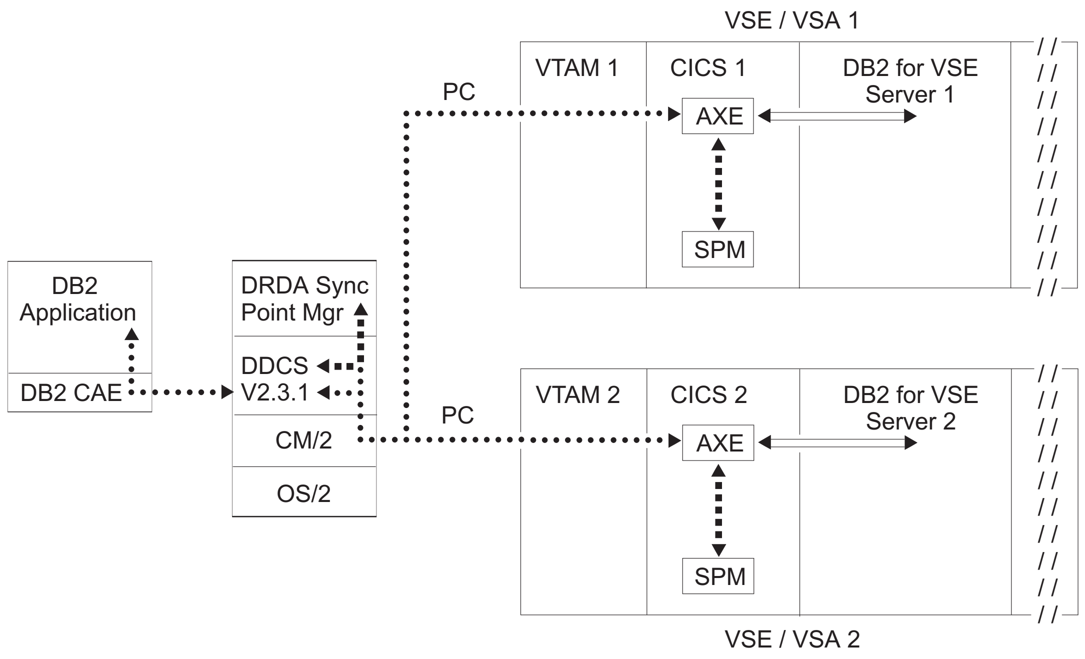


Figure 122. DB2 Server for VSE Distributed Unit of Work

This diagram shows how a workstation application might use DDCS for OS2 V2.3.1 to execute a distributed unit of work between 2 DB2 Server for VSE servers on different VM/ESA systems. Notice that DDCS registers itself with its own sync point manager. DDCS then establishes protected conversations with CICS AXE transactions. Each AXE transaction registers itself with a locally installed CICS/VSE sync point manager and CICS will perform any sync point logging and resynchronization activity. The AXE transactions use the VSE XPCC protocol to communicate with the DB2 Server for VSE servers 1 and 2.

DB2 Server for VSE also requires the use of a task-related user exit (TRUE) to interface with the CICS/VSE syncpoint manager and with the DB2 Server for VSE database manager as it participates in a coordinated commit or backout process. This interface is described in the *CICS/VSE Customization Guide*.

A separate task-related user exit program is enabled and started for each DB2 Server for VSE application server in support of distributed unit of work (two-phase commit). This is done by the AXE transaction under the following circumstances:

- DB2 Server for VSE database start-up parameter SYNCPNT=Y, and
- The conversation with the application requester is a **protected** basic conversation (synclevel 2), and
- The application starts doing some work (for example, begins using an agent), and
- The task-related user exit is not yet enabled.

The entry name of the task-related user exit program is 'ARI0*x*AXE', where *x* is 0->9, A->Z. *x* is the last character of a DB2 Server for VSE application server APPLID. The APPLID of a DB2 Server for VSE application server can be any of the following reserved DBNAMES:

- SYSARI0*0* to SYSARI0*9*,
- SYSARI0*A* to SYSARI0Z.

Therefore, the corresponding entry name of the task-related user exit program is

- ARI0*0*AXE to ARI0*9*AXE,
- ARI0*A*AXE to ARI0ZAXE.

CICS transactions for managing DRDA task-related user exits:

DAXP The **DAXP** transaction is a CICS transaction that sets the parameters that will be used when an AXE transaction subsequently autostarts DRDA TRUE (task-related user exit) support for an application server.

For more information on DAXP, see the *DB2 Server for VSE & VM Operation* manual.

DAXT The **DAXT** transaction is a task-related user exit administration routine that is responsible for disabling DRDA support within CICS/VSE. It issues the EXEC CICS DISABLE command to disable a task-related user exit program.

For more information on DAXT, see the *DB2 Server for VSE & VM Operation* manual.

Managing In-Doubt LUW's

Operator Commands

The following DB2 Server for VSE operator commands can be used to manage in-doubt LUWs:

SHOW ACTIVE
Displays the status of active agent structures[1]

SHOW CONNECT
Displays the status of all users or selected users connected to the application server.

FORCE COMMIT
Heuristically forces an in-doubt LUW to COMMIT

FORCE ROLLBACK
Heuristically forces an LUW to ROLLBACK

SHOW INDOUBT
Displays the status of all DRDA2 distributed units of work that are:
- currently in-doubt
- were heuristically committed or rolled back but RESYNC has not yet been performed nor has RESET INDOUBT been performed
- RESYNC failed for some in-doubt unit of work, because the status of the LUW was the opposite of what RESYNC required. (for example, RESYNC required that the unit of work be COMMITTED, but it had been heuristically ROLLED BACK). At the same time, RESET INDOUBT has not been performed.

RESET INDOUBT
Causes a heuristically committed or rolled back unit of work to be forgotten by the database, (that is, causes a *forget* log record to be written.)

Making Heuristic Decisions

In a DRDA environment, resynchronization occurs if two-phase commit processing is interrupted by a resource failure. However, the decision to commit or roll back an in-doubt LUW by any way other than the normal resynchronization process is a *heuristic decision*. If you commit or roll back a unit of work and your decision is different from the other system's decision, data inconsistency occurs. This type of damage is called *heuristic damage*.

An example of heuristic damage would be if the operator performed a heuristic commit and then the transaction manager requested that the unit of work be rolled back. If this situation occurs, and your system then updates any data involved with the previous unit of work, your data is corrupted and is difficult to correct.

The only way to correct heuristic damage is to restore the database from an archive by manually correcting the data based on knowledge from the application. This damage correction must be coordinated with all of the participating application servers to ensure that the data is consistent in each individual application server and between all of the participating application servers.

You can perform heuristic actions on in-doubt transactions that are not involved in a distributed unit of work. The heuristic actions performed are not logged and therefore are also not displayed by the SHOW INDOUBT operator command. (Note that heuristic damage is still possible on these transactions.) See the *DB2 Server for VSE & VM Operation* manual for more information on the SHOW INDOUBT command.

1. An agent is the internal database manager representation of an active user

Performing heuristic actions on distributed unit of work transactions must be done with caution. You can use the FORCE command to perform heuristic functions on distributed unit of work in-doubt LUWs when the resource owner cannot wait for the sync point manager to perform the resynchronization action. See the *DB2 Server for VSE & VM Operation* manual for more information on the FORCE command.

Resynchronization

Resynchronization occurs if two-phase commit processing is interrupted by a resource failure. A resource failure may be caused by a node failure, a session failure, a program failure or other problems by a protected resource manager. The resource failure may be between a sync point manager and local resource managers or sync point manager and remote resource managers.

Resynchronization is conducted independently for each failed protected resource for which it is required. Resynchronization has the following purposes:

- To place distributed resources in consistent states, if possible; if not possible, to notify the operator at the LU that detected the damage and at the LU of the root of the sync point tree. The LU for DB2 Server for VSE is CICS/VSE.
- To unlock locked resources in order to free them for other uses
- To update the log showing that no more sync point work is needed for that protected resource, for that LUW.

Resync When Partner is Not Active

After an LU failure, it is possible that the partner that is responsible for resync is unable to establish the resync conversation because the failed LU has not been restarted. The responsible LU retries the resync at implementation-defined intervals.

In order to reduce the delay for resynchronization after an LU is restarted, the partner LU may signal to the resync initiator that it is available by sending an Exchange Log Names GDS variable that is **not** accompanied by a Compare States GDS variable.[2] Once the responsible LU has received this signal that the failed LU is active, it can initiate resync, sending the Exchange Log Names and Compare States GDS variables.

Sending the Exchange Log Names GDS variable as a signal of LU availability need be done only once, no matter how many protected conversations require resynchronization between the two LUs. Also, if the LU that becomes available is responsible for initiating resync for some conversations, it need not send another Exchange Log Names GDS variable as a signal that the LU is available, since the partner SPM can infer that a partner is available from the other resyncs the partner SPM initiates.

Resolution of In-doubts

In the VSE environment, resynchronization has two components:

2. The partner can tell that a Compare States GDS variable is not present because SPM's RECEIVE_AND_WAIT verb will complete with a WHAT_RECEIVED of SEND rather than DATA_COMPLETE.

- LU6.2 resynchronization[3] occurs between the CICS/VSE syncpoint manager and the application requester syncpoint manager. This is initiated at the application requester site.

  If there is a failure of the APPC session with the source system during the in-doubt period, the CICS/VSE syncpoint manager makes a heuristic decision based on the **INDOUBT** [4] option of the AXE transaction definition.
- Task-related user exit resynchronization[5] occurs between the CICS/VSE syncpoint manager and the CICS external resource manager (DB2 Server for VSE). It is driven by the EXEC CICS RESYNC request that is issued by the AXE transaction after it has successfully enabled and started the DRDA TRUE. This causes the CICS/VSE syncpoint manager to pass the appropriate operation code (for example, commit or rollback) to the DRDA task-related user exit (TRUE) program for each in-doubt LUW that needs to be resynchronized. Depending on the nature of the operation code, the DRDA TRUE issues an SQL request to the DB2 Server for VSE database manager to resolve the in-doubt LUW.

The details of what the task-related user exit does when it receives the operation code from the CICS/VSE Syncpoint Manager during task-related user exit resynchronization are shown in Table 35.

Table 35. Actions by the DRDA task-related user exit during task-related user exit resynchronization

CICS/VSE Syncpoint Manager operation code received by TRUE	**DRDA task-related user exit actions**
Backout (UERTBACK)	Issues EXEC SQL ROLLBACK to resource manager
Committed (UERTCOMM)	Issues EXEC SQL COMMIT to resource manager

Lost due to CICS cold start (UERTDGCS)	Generates a message indicating that the LUW is lost due to CICS cold start:
	`ARI0193E The CICS log does not contain information for an in-doubt`
	`logical unit of work belonging to server` *`server_name`*`.`
	
	Instructs the operator to FORCE the LUW if the LUW has an agent assigned to it (for example, the LUW has not been FORCED):
	`ARI4018A Use the database manager SHOW and FORCE commands`
	`to commit or rollback the following units of work:`
	`ARI4019I SERVER_NAME =` *`server_name`*`.`
	`USER ID =` *`user id`*`.`
	`AGENT IDENTIFIER =` *`N`*`.`
	
	Instructs the operator to RESET INDOUBT if the LUW does not have an agent assigned to it (for example, the LUW has been FORCED):
	`ARI0195A Use the database manager SHOW INDOUBT operator`
	`command to find the following units of work:`
	`ARI0196I SERVER_NAME =` *`server_name`*`.`
	`RECOVERY TOKEN =` *`rectok`*`.`

3. For general information on how CICS handles this, see the section "Syncpoint and Recovery" in the "VTAM LU6.2" chapter of the *CICS/VSE Diagnosis Reference* manual

4. For more information on the INDOUBT option, see the section ″The INDOUBT Option of the Transaction Definition″ in the ″Recovery and Restart in Interconnected Systems″ chapter of the *CICS/VSE Intercommunication Guide*.

5. For general information on how CICS handles user exit resynchronization, see the chapter ″Task-related User Exit Recovery″ in the *CICS/VSE Diagnosis Reference* manual

Table 35. Actions by the DRDA task-related user exit during task-related user exit resynchronization (continued)

CICS/VSE Syncpoint Manager operation code received by TRUE	**DRDA task-related user exit actions**

LUW should not be in doubt (UERTDGNK)	Generates a message indicating that the LUW should not be in doubt:
	`ARI0194E A logical unit of work that the database manager`
	`         for server *server_name* indicated needed to be`
	`         resolved was not identified by the CICS/VSE log as needing`
	`         resolution.`
	Instructs the operator to FORCE the LUW if the LUW has an agent assigned to it (for example, the LUW has not been FORCED):
	`ARI4018A Use the database manager SHOW and FORCE commands`
	`         to commit or rollback the following units of work:`
	`ARI4019I SERVER_NAME = *server_name*.`
	`         USER ID = *user id*.`
	`         AGENT IDENTIFIER = *N*.`
	Instructs the operator to RESET INDOUBT if the LUW does not have an agent assigned to it (for example, the LUW has been FORCED):
	`ARI0195A Use the database manager SHOW INDOUBT operator`
	`         command to find the following units of work:`
	`ARI0196I SERVER_NAME = *server_name*.`
	`         RECOVERY TOKEN = *rectok*.`

Note: 1. When the SHOW and FORCE operator commands are used to commit or rollback an LUW, the RESET INDOUBT operator command must be issued to cause DB2 Server for VSE to forget about the LUW. 2. When message ARI0195A is issued, the RESET INDOUBT operator command must be issued to cause DB2 Server for VSE to forget about the LUW. 3. In both cases, before RESET INDOUBT is issued, any user-defined action to resynchronize the local and remote databases must be done. 4. VSE messages come up in the CICS partition, not in the partition you are working on.	

The DB2 Server for VSE resource manager processes the SQL request it received from the task-related user exit. The action it takes is determined by the SQL request and the LUW state which it remembers. These actions are shown in Table 36.

Table 36. Actions by the DB2 Server for VSE resource manager during task-related user exit resynchronization

LUW state at DB2 Server for VSE	**SQL request passed by task-related user exit**	
	Rollback	**Commit**

LUWID Not Found	Send normal completion reply indicating Backout state. DB2 Server for VSE notifies operator with message:
	`ARI0183E The Sync Point Manager has asked to *ROLLBACK*`
	`         this LUW but the database manager has no`
	`         memory of it.`
	`ARI0196I SERVER_NAME = *server_name*.`
	`         RECOVERY TOKEN = *rectok*.`
		Send normal completion reply indicating Committed state. DB2 Server for VSE notifies operator with message:
		`ARI0183E The Sync Point Manager has asked to *COMMIT*`
		`         this LUW but the database manager has no`
		`         memory of it.`
		`ARI0196I SERVER_NAME = *server_name*.`
		`         RECOVERY TOKEN = *rectok*.`

Indoubt (Prepared)	Drive backout of resource and send normal completion reply indicating Backout state.	Drive commit of resource and send normal completion reply indicating Committed state.

Table 36. Actions by the DB2 Server for VSE resource manager during task-related user exit resynchronization (continued)

LUW state at DB2 Server for VSE	**SQL request passed by task-related user exit**	
	Rollback	**Commit**

Heuristic Backout	Send normal completion reply indicating Backout state.	Send normal completion reply indicating Committed state. DB2 Server for VSE notifies operator with message:
		`ARI0184A The Sync Point Manager has asked to COMMIT this LUW but the FORCE command was previously used to ROLLBACK it.`
		`ARI0196I SERVER_NAME = server_name. RECOVERY TOKEN = rectok.`
		In this case, the LUW will still appear when the SHOW INDOUBT command is executed. The LUW must be cleared using the RESET INDOUBT command. In addition, manual intervention is necessary to ensure that the LUW is in a consistent state at all sites where the LUW has been distributed. This may require intervention at this database manager, or possibly at another database manager. Manual intervention could mean manually fixing the data or possibly restoring an archive.

Heuristic Committed	Send normal completion reply indicating Backout state. DB2 Server for VSE notifies operator with message:
	`ARI0184A The Sync Point Manager has asked to ROLLBACK this LUW but the FORCE command was previously used to COMMIT it.`
	`ARI0196I SERVER_NAME = server_name. RECOVERY TOKEN = rectok.`
	In this case, the LUW will still appear when the SHOW INDOUBT command is executed. The LUW must be cleared using the RESET INDOUBT command. In addition, manual intervention is necessary to ensure that the LUW is in a consistent state at all sites where the LUW has been distributed. This may require intervention at this database manager, or possibly at another database manager. Manual intervention could mean manually fixing the data or possibly restoring an archive.
		Send normal completion reply indicating Committed state.

Table 36. Actions by the DB2 Server for VSE resource manager during task-related user exit resynchronization (continued)

LUW state at DB2 Server for VSE	**SQL request passed by task-related user exit**	
	Rollback	**Commit**

Note:

1. The state **Syncpoint Pending** is not possible at DB2 Server for VSE servers. The server completes any sync point actions such as prepare to commit, commit, or rollback before the CICS/VSE Syncpoint Manager performs any sync point logging.
2. The state **Backout (Reset)** is not possible at DB2 Server for VSE servers. The servers complete rollback processing before the CICS/VSE Syncpoint Manager performs any sync point logging for backout.
3. The state **committed** is not possible at DB2 Server for VSE servers. The servers complete commit processing before the CICS/VSE Syncpoint Manager performs any sync point logging for committed.
4. It is very remote that "LUWID Not Found" would occur. It can only happen if the following occurs:
   a. TRUE enablement support obtains from the DB2 Server for VSE resource manager a list of LUWIDs that require resynchronization.
   b. The FORCE and RESET INDOUBT operator commands are issued for an LUWID that was in the list of LUWIDs above.
   c. TRUE enablement support commits or backs out LUWID according to the disposition of the CICS URD.
5. The task-related user exit resynchronization process is all one way, from CICS to the external resource manager (such as, DB2 for VSE). After the task-related user exit has obtained a list of in-doubt LUWs from the DB2 for VSE resource manager and has passed this list to CICS via the RESYNC command, CICS looks at its URDs to determine whether to commit or backout each LUW. There is no provision for a DB2 for VSE resource manager, through its task-related user exit, to inform CICS that a heuristic decision was made for an LUW.

   The only sensible thing for the DB2 Server for VSE resource manager to do, in the case of a heuristic damage, is to send a normal completion reply, as opposed to sending an abnormal reply. This way, the task-related user exit can vote UERFDONE (Forget) to the CICS/VSE Syncpoint Manager and the CICS/VSE Syncpoint Manager can throw away the unit of recovery descriptor (URD) associated with this LUW. If the DB2 Server for VSE resource manager sends an abnormal reply, the task-related user exit will vote UERFHOLD and CICS will hold the URD for this LUW until the next RESYNC. If the resource manager had ″forgotten″ the LUW, (for example, RESET INDOUBT was done), CICS will assume that the resource manager is not interested in this LUW and will throw away the corresponding URD. Therefore, both normal and abnormal replies eventually produce the same results, but it is more efficient to send a normal reply in the case where a heuristic damage has occurred.

Chapter 15. Using TCP/IP with DB2 Server for VSE

TCP/IP communications can be used with DB2 Server for VSE using DRDA protocol. DB2 Server for VSE application requesters can use DRDA remote unit of work over TCP/IP to access remote DRDA-capable servers (including remote DB2 Server for VSE and DB2 Server for VM servers.) Non-DB2 Server for VSE requesters can use DRDA remote unit of work over TCP/IP to access remote DB2 Server for VSE servers (including remote DB2 Server for VSE and DB2 Server for VM requesters).

Preparing the Application Server to use TCP/IP

The following must be done to allow the application server to use TCP/IP.

1. TCP/IP for VSE must be installed and configured.
2. The LE/VSE runtime libraries must be available. For the Batch Application Requester, this is the PRD2.SCEEBASE library. For the Online Application Requester on VSE/ESA Version 2 Release 3 and 4, these are the PRD2.SCEECICS and the PRD2.SCEEBASE libraries. For the Online Application Requestor on VSE/ESA Version 2 Release 5 and later, it is the PRD2.SCEEBASE library. Minimum support level of LE/VSE is Version 1 Release 4.
3. The TCP/IP for VSE library must be available on both Online and Batch application requesters, this is the PRD1.BASE library. If your TCP/IP product is ordered directly from CSI (Connectivity Systems) or from a distributor, the product library is PRD2.TCPIP.
4. To ensure proper TCP/IP functionality, the C runtime library and the TCP/IP library must follow a certain search sequence. The following are sample LIBDEF statements for the possible combinations:
   - TCP/IP product installation from IBM:

     Batch partition:

     ```
     //LIBDEF*,SEARCH=(PRD1.BASE,PRD2.SCEEBASE,...)
     ```

     Online partition:

     – For VSE/ESA Version 2 Release 3 and 4:

     ```
     //LIBDEF*,SEARCH=(PRD1.BASE,PRD2.SCEECICS,PRD2.SCEEBASE,...)
     ```

     – For VSE/ESA Version 2 Release 5 and later:

     ```
     //LIBDEF*,SEARCH=(PRD1.BASE,PRD2.SCEEBASE,...)
     ```

   - TCP/IP product installation from CSI:

     Batch partition:

     ```
     //LIBDEF*,SEARCH=(PRD2.TCPIP,PRD1.BASE,PRD2.SCEEBASE...)
     ```

     Online partition:

     – For VSE/ESA Version 2 Release 3 and 4:

     ```
     //LIBDEF*,SEARCH=(PRD2.TCPIP,PRD1.BASE,PRD2.SCEECICS,PRD2.SCEEBASE,...
     ```

     – For VSE/ESA Version 2 Release 5 and later:

     ```
     //LIBDEF*,SEARCH=(PRD2.TCPIP,PRD1.BASE,PRD2.SCEEBASE,...
     ```

5. One JCL statement //OPTION SYSPARM='*xx*' should be added to the CICS startup JCL or the batch JCL, where *xx* is to match the xx specified in the ID=xx parameter for starting up a specific TCP/IP server. This is how the application requesters can route the TCP/IP function request to the correct TCP/IP server

in case there are more than one TCP/IP server running in the same VSE/ESA system. If //OPTION SYSPARM is not specified, the default is 00.

6. The VSE partition running the TCP/IP for VSE server should always have a higher priority than the partition running the DB2 for VSE database server.

TCP/IP support is invoked at system initialization time. If TCP/IP for VSE is available, the server will make use of it. The application server must be able to determine what port number to listen on for connections. This can be accomplished in a number of ways.

1. The DBNAME Directory of the database has a port number specified for the TCPPORT parameter in the application server's DBNAME Local entry. See "Setting Up the DBNAME Directory" on page 23.
2. The new initialization parameter, TCPPORT, can be used to specify the port number to listen on. Refer to the *DB2 Server for VSE & VM Operation* manual for a detailed description of the TCPPORT parameter.
3. The well-known port number 446 is used, if available.

Each method has advantages and disadvantages.

The first method of using the DBNAME Directory is the preferred method. This directory is maintained by the database administrator and resides in a VSE library. Since more than one DB2 Server for VSE can run on the same VSE system, it must be ensured that they do not use the same TCP/IP port or users will be connected to the wrong database. Identifying the port numbers in the directory makes it easier to ensure that different servers are using different ports.

The second method of port identification is the new initialization parameter, TCPPORT. This is helpful when initially testing TCP/IP support or when TCP/IP support needs to be enabled, but the DBNAME Directory cannot be updated. The disadvantage is that it is possible that another application may be using the same port. If this occurs, an error message is received during initialization showing a BIND failure with return code 1115 indicating that the port was already in use by another application.

The third method is the least desirable. If there is no port number specified in the DBNAME Directory for the application server or a TCPPORT initialization parameter was not specified, there is a well known port assignment for relational databases. It is called ddm-rdb and the port number is 446. This has the advantage of doing no extra configuration to TCP/IP for VSE and to the application server. The disadvantage is that only one application server on the VSE system can use the definition.

We will take advantage of all of the methods. The actions can be broken down into the following scenarios.

1. If the TCPPORT initialization parameter is not specified when the application server is started, the application server will search the DBNAME Directory for its corresponding Local entry to see if the TCPPORT parameter is specified. If it is specified and the value is not zero, it will be used as the port number of the listener socket to be created. If the value is zero, no TCP/IP initialization will be performed. If the TCPPORT parameter is not specified, it will use the well known port number 446 to create the listener socket.
2. If the TCPPORT is specified when the application server is started, the application server will use this parameter while performing TCP/IP support initialization. If the port cannot be used, no attempt is made to find another

port. If any error is returned from a TCP/IP function used, it will be assumed that TCP/IP is not available and TCP/IP support on the application server will not be used.

After the TCP/IP support for the application server is initiated, a TCP/IP agent is created to handle all TCP/IP related functions. If any TCP/IP function failure is detected by the TCP/IP agent, the TCP/IP support for the application server will be disabled. It is possible to restart the TCP/IP support for the application server without recycling the application server. This can be done by using the START TCPIP operator command. Refer to the *DB2 Server for VSE & VM Operation* manual for a detailed description of this command. The restart will also be done automatically by the database manager if the TCPRETRY parameter is set to Y.

To have the database manager re-enable TCP/IP support automatically, the database manager must have the TCPRETRY initialization parameter set to Y. This can be done by specifying the initialization parameter TCPRETRY=Y when the database manager is initialized or by using the operator command SET TCPRETRY Y. The default value for this parameter is Y. The current setting of the parameters can be checked with the SHOW INITPARM operator command.

If TCP/IP support fails and automatic restart is enabled, the database manager will use the following strategy to restart TCP/IP support.

1. The database manager will try to re-enable TCP/IP support every 30 seconds up to 10 times. The maximum recovery time is 5 minutes.
2. If TCP/IP is still disabled, the database manager will try to re-enable TCP/IP support every 60 seconds up to 5 times. The maximum recovery time is now up to 10 minutes.
3. If TCP/IP is still disabled, the database manager will try to re-enable TCP/IP support every 10 minutes. It will do this until the number of attempts is greater than the TCPMAXRT value or until TCP/IP support is re-enabled or the operator disables the retry attempts.

To disable automatic retry, issue the operator command SET TCPRETRY N. If automatic retry is not desired at all, specify the initialization parameter TCPRETRY=N when the database manager is initialized. If TCPRETRY is set back to Y after being set to N, the number of attempts and the retry interval are reset to their original values. The number of attempts is set to 0 and the first retry interval is set to 30 seconds. If TCP/IP support is disabled when this occurs, then automatic retry will begin in 30 seconds.

If the maximum number of retry attempts is reached and TCP/IP support has not been successfully re-enabled, the database manager will disable the automatic restart support by setting the value of TCPRETRY to N. The maximum number of retry attempts is controlled by the TCPMAXRT parameter. This parameter can be specified at database initialization and it can be modified online by the SET command. The default value is set to 158. This results in 24 hours of retry attempts. The maximum value is 9,999 which results in 69 days of retry attempts. The following formulas are useful for determining a suitable value for TCPMAXRT. These formulas are only valid for values of TCPMAXRT that are greater than or equal to 15 or time values greater than or equal to 10 minutes.

Given a value for TCPMAXRT, how long will retry be attempted?

minutes – 10 + ((TCPMAXRT – 15) * 10)

Given a time limit in minutes, what should TCPMAXRT be set to?

TCPMAXRT = 15 + ((minutes − 10) / 10)

Using the first formula, we can see that a value of 158 for TCPMAXRT would take 10 + (158−15)*10 or 1440 minutes. 1440 minutes is equivalent to 24 hours. To enable the retry for 2 days we use the second formula. 2 day is 2 * 24 * 60 or 2880 minutes. 15 + ((2880-10)/10) = 15 + (2870/10) = 15 + 287 = 302. Setting TCPMAXRT to 302 will result in retry being attempted for 2 days.

Preparing the Application Requester to use TCP/IP

The following must be done to allow an application requester to use TCP/IP.

1. TCP/IP for VSE must be installed and configured.

To indicate that TCP/IP is to be used to establish a connection from the online application requester, the SQLGLOB file is used. If T is specified as the communication protocol for the referencing application, TCP/IP communication protocol will be used. In this case, the DBNAME Directory remote entry of the database that is the target of the SQL CONNECT statement of the application must be set up with the necessary TCP/IP related information to be used to establish a TCP/IP connection. Refer to “Setting Up the DBNAME Directory” on page 23 for details. For more information about the SQLGLOB file, refer to the *DB2 Server for VSE & VM Database Administration* manual.

Note that batch applications accessing remote servers will always use TCP/IP protocol and the SQLGLOB file communications protocol parameter is ignored.

Appendix A. Processor Storage Requirements

(The information in this Appendix has been moved to the *DB2 Server for VSE Program Directory*.)

Appendix B. Estimating Database Storage

This appendix describes procedures for estimating the size of the directory, the SYS0001 dbspace and ISQL dbspace.

For information on estimating the size of user dbspaces, see the *DB2 Server for VSE & VM Database Administration* manual.

Storage Capacities of IBM DASD Devices

The effective storage capacities of IBM DASD devices vary, depending on how the devices are being used. The database manager uses VSE/VSAM for managing DASD space for the directory data set, the log, and the dbextents. The directory data set uses 512-byte control intervals while the log and dbextent data sets are managed with 4-kilobyte control intervals.

Table 37 and Table 38 show the capacities of IBM devices for storing log and dbspace pages (dbextent space). Table 39 and Table 40 show the capacities for storing directory information.

Table 37. Log and Dbextent Storage Capacities of IBM Count-Key-Data DASDs

DASD Type	Number of Cylinders	Tracks for Each Cylinder	Megabytes for Each Cylinder	Megabytes for Each Volume
3375	959	12	0.3749	359
3380 J	885	15	0.5858	518
3380 E	1,770	15	0.5858	1,036
3380 K	2,655	15	0.5858	1,555
3390-1	1,113	15	0.7031	782
3390-2	2,226	15	0.7031	1,565
3390-3	3,339	15	0.7031	2,347
3390-9	10,017	15	0.7031	7,041
9345-1	1,440	15	0.5858	843
9345-2	2,156	15	0.5858	1,262

Table 38. Log and Dbextent Storage Capacities of IBM FBA DASDs

DASD Type	Megabytes for Each Volume	4 Kilobyte Pages for Each Volume
3370-1	272.4	69,750
3370-2	348.0	89,094
9332-400	175.7	45,004
9332-600	270.8	69,350
9335	392.9	100,589
9336-010	449.2	115,014
9336-020	816.8	209,110
9336-025	816.8	209,110

Table 38. Log and Dbextent Storage Capacities of IBM FBA DASDs (continued)

DASD Type	Megabytes for Each Volume	4 Kilobyte Pages for Each Volume
0671	280.5	71,820

Table 39. Directory Storage Capacities of IBM Count-Key-Data DASDs

DASD Type	Number of Cylinders per Volume	Tracks for Each Cylinder	Megabytes for Each Cylinder	Megabytes for Each Volume
3375	959	12	0.2343	224
3380 J	885	15	0.3295	291
3380 E	1,770	15	0.3295	583
3380 K	2,665	15	0.3295	874
3390-1	1,113	15	0.3645	405
3390-2	2,226	15	0.3645	811
3390-3	3,339	15	0.3645	1,217
3390-9	10,017	15	0.3645	3,651
9345-1	1,140	15	0.3002	432.28
9345-2	2,156	15	0.3002	647.23

Table 40. Directory Storage Capacities of IBM FBA DASDs

DASD Type	Megabytes for Each Volume	512-Byte Blocks for Each Volume
3370-1	272.4	558,000
3370-2	348.8	712,752
9332-400	175.7	360,036
9332-600	270.8	554,800
9335	392.9	804,714
9336-010	449.1	920,115
9336-020	816.8	1,672,881
9336-025	816.8	1,672,881
0671	280.5	574,560

These capacity charts are referenced in later calculations for determining data set allocations of the directory, log, and dbextent data sets.

Table 41 shows the minimum space allocations for a log or dbextent data set.

Table 41. Minimum Space Allocations for Log and Dbextent Data Sets

DASD Type	Minimum Space Allocation
3375	1 cylinder
3380	1 cylinder
3390	1 cylinder
9345	1 cylinder
FBA	528 Blocks

Relationship of Megabytes to 4-Kilobyte Pages

In the database generation process, all dbspace and dbextent DASD space definitions are expressed in terms of 4-kilobyte pages: that is, each page represents 4096 bytes of storage space. Storage space is used not only for data, but also for indexes and free space initially reserved to facilitate the insertion of new data after the database is in operation.

Space needs are often expressed in terms of megabytes (1,048,576 bytes). Table 42 shows the number of 4-kilobyte pages needed to support a range of megabytes. The dbspace definitions are made in multiples of 128 pages. An alternative to using Table 42 is to use the formula:

```
Number of 4-kilobyte pages  =  256 x number of megabytes
```

Table 42. Megabytes of Data on 4-Kilobyte Pages

Megabytes	4-Kilobyte Pages
0.0 - 0.5	128
0.5 - 1.0	256
1.0 - 1.5	384
1.5 - 2.0	512
2.0 - 2.5	640
2.5 - 3.0	768
3.0 - 3.5	896
3.5 - 4.0	1,024
4.0 - 4.5	1,152
4.5 - 5.0	1,280
5.0 - 5.5	1,408
5.5 - 6.0	1,536
6.0 - 6.5	1,664
6.5 - 7.0	1,792
7.0 - 7.5	1,920
7.5 - 8.0	2,048
8.0 - 8.5	2,176
8.5 - 9.0	2,304
9.0 - 9.5	2,432
9.5 - 10.0	2,560
50.0	12,800
100.0	25,600
500.0	128,000

Estimating Directory Space Requirements

The required size of the database directory depends on the maximums you established on the MAXPOOLS, MAXEXTNT, and MAXDBSPC parameters during database generation. The directory must be large enough to hold page table entries for the maximum size of the database. Figure 123 shows a formula for calculating the recommended size of a directory data set.

```
Directory size = 7 558 + 16      x MAXDBSPC value
  (in bytes)          + 16      x MAXEXTNT value
                      +  4      x MAXPOOLS value
                      +  0.0021 x Maximum database size
```

Figure 123. Formula for Calculating Directory Size (in Bytes)

To estimate the value for the maximum database size, determine how many dbspaces (public, private, and internal) your database will need, and the number of pages needed by each dbspace; then multiply the total number of pages by 4096 to get the number of bytes. (You may want to overestimate this value to allow for creating unplanned dbspaces, and for increasing the number and size of internal dbspaces.) Finally, multiply this number by 0.0021, to determine how many bytes are needed in the directory to support these dbspaces. The result of this calculation includes the space needed for shadow paging.

Once you have the directory size, you can use the charts shown in the section "Storage Capacities of IBM DASD Devices" on page 341 to determine the data set size specifications, in cylinders or blocks, of the device to be used.

Note: Although you do not have to specify the maximum database size during database generation, the size specified for the directory data set effectively establishes the limit.

Estimating Storage Pool Requirements

For estimating storage pool sizes, you need to estimate:

- The size of used portions of dbspaces. This includes tables, indexes, and free space on used dbspace pages.
- Shadow paging requirements. This is an estimate of the number of dbspace pages that can change between checkpoints.

To estimate the number of pages required for a storage pool use the following formula:

```
Pool pages = 8 x Number of dbspaces
           + 1.5 x Data pages for all dbspaces in the pool
           + Data pages for the largest table in the pool
```

This calculation covers header pages and pages required for table rows and indexes on those tables. If you have increased your dbspace data pages value to accommodate future growth of tables, you can decrease the pool pages correspondingly.

The addition of the factor of data pages for the largest table in the pool should accommodate storage pool demands for shadow paging. This allows for UNLOAD and RELOAD of the largest table in the storage pool.

Estimating SYS0001 Dbspace Requirements

The PUBLIC.SYS0001 dbspace is reserved for the catalog tables during database generation, and cannot be redefined. You establish its size (and storage pool) when you generate the database. The size should be large enough to hold all of your database catalog information for the life of the database.

Note: Physical space is not actually consumed until it is required. Consequently, you can define the SYS0001 dbspace to be very large without penalty. Be generous. The penalty for defining the SYS0001 dbspace too small is that, when it has no more usable space, you must completely regenerate the database. This can be a considerable task for a production database. For more information, see "Preparing for Database Regeneration" on page 34.

The formula shown in Figure 7 on page 20 should provide ample storage space for most uses of the database manager. The formula was derived based on a set of assumptions that may not be valid for your database. Review the assumptions and modify the general formula if the assumptions do not accurately represent your planned usage of the database manager.

The following sections describe:

- SYS0001 storage estimating general formula assumptions

  You should review these assumptions to determine whether they apply for your planned usage of the database manager. If they do not, you should modify the assumptions (and the resulting formula) to more accurately represent your planned usage.
- Derivation of the general formula for SYS0001 storage estimating
- Formula for SYS0001 storage estimating

  This formula is described in "Formula for SYS0001 Storage Estimating" on page 349.
- Examples of using the SYS0001 storage estimating formula

  These examples show how to use the SYS0001 storage estimating formula based on three example situations.
- Modifying the SYS0001 storage estimating general formula

  This section provides the formulas used to derive the general formula. You can modify the general formula if you want to change some of the assumptions made in deriving the general formula.

SYS0001 Storage Estimating General Formula Assumptions

The general formula for SYS0001 storage estimating was derived based on:

- Average row lengths for catalog rows
- The number of rows required for each object type in the formula.

Average Row Lengths for Catalog Table Rows

Table 43 on page 346 shows the length of the fixed portions of catalog rows, the maximum stored row length for each catalog table, and an average row length for each of the catalog tables. The average row length is the length assumed in developing the general formula for estimating catalog storage space requirements.

Table 43. Stored Lengths of Catalog Rows

Catalog Table	Minimum Length	Maximum Length	Estimated Average Length
SYSACCESS	46	90	52
SYSCATALOG	64	385	170
SYSCCSIDS	39	39	39
SYSCHARSETS	393	411	400
SYSCOLAUTH	46	82	72
SYSCOLSTATS	27	123	59
SYSCOLUMNS	54	398	156
SYSDBSPACES	40	58	46
SYSDROP	13	13	13
SYSINDEXES	62	232	131
SYSKEYCOLS	55	91	67
SYSKEYS	77	113	89
SYSOPTIONS	11	301	100
SYSPARMS	82	82	82
SYSPROGAUTH	46	54	49
SYSPSERVERS	11	281	60
SYSROUTINES	58	581	170
SYSSTRINGS	286	286	286
SYSSYNONYMS	26	62	36
SYSTABAUTH	57	101	84
SYSUSAGE	36	72	51
SYSUSERAUTH	35	35	35
SYSVIEWS	20	293	200

In Table 43, the minimum and maximum row lengths for each catalog table are determined using the description of the catalog tables in the *DB2 Server for VSE & VM Database Administration* manual. The length of a row depends on the data type of each column in the catalog table. The minimum length for each column is found using these values for each data type.

Table 44. Minimum Column Length

Data Type	Value
DBAINT	4
DBAHW	2
INTEGER	4
SMALLINT	2
CHAR(*n*)	*n*
TIMESTAMP	10
VARCHAR(*n*)	1

Note: The data types DBAINT and DBAHW are used internally by the database manager. Externally, they look like the data types INTEGER and SMALLINT.

For CHAR columns, the length is the column length (*n*). The column lengths are added. For each column that can contain nulls, 1 is added to this figure. The value 8 is then added to this total for catalog table overhead. The resulting number is the minimum row length for the catalog table.

The maximum length for each column is found using these values:

Table 45. Maximum Column Length

Data Type	Value
DBAINT	4
DBAHW	2
INTEGER	4
SMALLINT	2
CHAR(*n*)	*n*
TIMESTAMP	10
VARCHAR(*n*)	*n* + 1

For CHAR columns, the length is the column length (*n*). For VARCHAR columns, the length is the maximum column length plus one (*n* + 1). For each column that can contain nulls, 1 is added to this figure. The value 8 is then added to this total for catalog table overhead. The resulting number is the maximum row length for the catalog table.

The average length for each column is calculated this way for most catalog tables:

$$\frac{\texttt{(maximum length - minimum length)}}{3} \quad + \quad \texttt{minimum length}$$

This produces a number one third of the way between the minimum and maximum lengths. In some situations, higher values are used because those columns are typically longer. An example is the SYSTEM.SYSVIEWS catalog table, where the VIEWTEXT column contains the command used to create the view. Because these commands are usually over 100 bytes long, a number one third of the way between the minimum and maximum lengths of the column would be too low. In this situation, the number 200 is chosen arbitrarily.

If you make your own estimates of catalog table row lengths (using the chart provided in Table 50 on page 352), you should choose values for the average row lengths that are accurate for your database. Otherwise, you could underestimate the size of the SYS0001 dbspace. In particular, you should not underestimate the average length of rows in the SYSTEM.SYSCOLUMNS catalog table. If you use the REMARKS or CLABEL columns of this catalog table, your average row length could be far greater than the number (156) given in Table 43 on page 346. Because the SYSTEM.SYSCOLUMNS table can become quite large (it has a row for every column in every table in the database), its size is a major factor in the size of the SYS0001 dbspace.

Assumptions on the Number of Catalog Table Rows

The average number of rows for each catalog table was determined based on the assumptions in Table 46. These assumptions were used in generating the general formula for SYS0001.

Table 46. Assumptions of Catalog Bytes/Pages for Each Object

Object	Catalog Entries	Bytes	Bytes for Each Object	Pages for Each Object
Table	1 SYSCATALOG 1 SYSTABAUTH 2 SYSINDEXES	169 84 262	515	0.13
View	1 SYSCATALOG 1 SYSVIEWS 2 SYSTABAUTH 2 SYSUSAGE	169 200 168 102	639	0.16
Column	1 SYSCOLUMNS	156	156	0.04
Package	1 SYSPROGAUTH 15 SYSUSAGE	49 765	814	0.20
Dbspace (including package dbspaces)	1 SYSDBSPACES	46	46	0.01
User	1 SYSUSERAUTH 50 SYSTABAUTH 50 SYSSYNONYMS 150 SYSCOLAUTH	35 4,200 1,800 10,800	16,835	4.11
Package dbspaces	255 SYSACCESS	13,260	13,260	3.24
Character Set	1 SYSCHARSETS	400	400	0.10
Keys	1 SYSKEYS 2 SYSKEYCOLS	89 134	223	0.05
Other	15 SYSOPTIONS	1200	1200	0.30

When a table is created, one entry is made in the SYSTEM.SYSCATALOG table and one in the SYSTEM.SYSTABAUTH table. This formula assumes an average of two indexes for each table. For each index created, one entry is made in SYSTEM.SYSINDEXES.

When a view is created, one entry is made in SYSTEM.SYSCATALOG. In addition, as many as 32 entries are made in SYSTEM.SYSVIEWS. With the assumption that the average view definition is less than 254 bytes, only one row is required. One entry is also made in the SYSTEM.SYSTABAUTH and SYSTEM.SYSUSAGE tables for each table on which the view is defined. The general formula assumes that, on average, a view is defined on two tables.

One entry is made in SYSTEM.SYSCOLUMNS for every table and view column.

When a package is created, one entry is made in SYSTEM.SYSPROGAUTH. In addition, entries are made in SYSTEM.SYSUSAGE for every table, view, index, and dbspace used by the package. (A package uses a dbspace if it uses a table in the dbspace.)

The general formula assumes 15 such entries in SYSTEM.SYSUSAGE.

One entry is made in SYSTEM.SYSDBSPACES for each dbspace added to the database, including package dbspaces.

One entry is placed in SYSTEM.SYSUSERAUTH for each user of the database. Each user is assumed to have access to an average of 50 tables (and views) belonging to other users. This explains the 50 entries in SYSTEM.SYSTABAUTH and SYSTEM.SYSSYNONYMS. Specific column update authorization is assumed to average about 3 columns for each table (or view) that is shared (3 for each of the 50 tables or views). This yields an estimate of 150 entries in SYSTEM.SYSCOLAUTH for each user.

For each package dbspace added, one entry is made in SYSTEM.SYSDBSPACES, which was accounted for earlier, and 255 entries are made in SYSTEM.SYSACCESS. The 255 entries are made because all 255 packages are preallocated in the dbspace, even though they can all be empty.

For each character set you define, you must load one row into SYSTEM.SYSCHARSETS.

For each key, one row is added to SYSTEM.SYSKEYS, and two rows are added to SYSTEM.SYSKEYCOLS (assuming that each key is made up of two columns).

Finally, three rows exist in SYSTEM.SYSOPTIONS for every database.

Derivation of the General Formula for SYS0001 Storage Estimating

The assumptions in the preceding section provide a means of estimating the data pages required in SYS0001. Assuming the PCTFREE value for the SYS0001 dbspace is 0, the SYS0001 data pages are:

```
SYS0001 data pages =    .13 x the number of tables
                     +  .16 x the number of views
                     +  .04 x the number of columns
                     +  .20 x the number of packages
                     +  .01 x the number of dbspaces
                              (including package dbspaces)
                     + 4.11 x the number of users
                     + 3.24 x the number of package dbspaces
                     +  .10 x the number of character sets
                     +  .05 x the number of keys
                     +  .30 (for the SYSTEM.SYSOPTIONS table)
```

To get the total number of SYS0001 dbspace pages, you must add the header pages and the index pages. SYS0001 has eight header pages. The initial set of catalog entries generated by the database generation process fills 4 pages. The PCTINDX value for SYS0001 is 60. Thus, to get the total number of pages you must add 12 and divide by 0.4:

```
SYS0001 pages = ( 12 + SYS0001 data pages ) / 0.40
```

The SYS0001 data pages is your estimate for the number of data pages for your catalog entries.

Formula for SYS0001 Storage Estimating

When the adjustments described in “Derivation of the General Formula for SYS0001 Storage Estimating” are made, the formula for the total number of SYS0001 dbspace pages becomes:

```
SYS0001 pages =   30 +   .33 x the number of tables
                     +   .40 x the number of views
                     +   .10 x the number of columns
                     +   .50 x the number of packages
                     +   .03 x the number of dbspaces
                               (including package dbspaces)
                     + 10.28 x the number of users
                     +  8.10 x the number of package dbspaces
                     +   .25 x the number of character sets
                     +   .13 x the number of keys
                   ( +   .74 (for the SYSTEM.SYSOPTIONS table)   )
```

This number should be rounded up to the next higher multiple of 128. Because the number of pages needed for the SYSTEM.SYSOPTIONS catalog table is so small, the number is omitted from the general formula and any further calculations.

Examples of Using the SYS0001 Storage Estimating Formula

The following examples illustrate the use of the general formula for estimating the required dbspace size for SYS0001.

For a Test Database

Table 47 illustrates the estimate for a small set of catalog tables that can be used in generating a test database.

Table 47. Example of Estimating the Catalog Dbspace for a Test Database

Example Number of Objects	Number of Pages Calculation	Number of Pages
Reserved	30	30
50 tables	.33 X 50	17
100 views	.40 X 100	40
1500 columns	.10 X 1 500	150
25 packages	.50 X 25	13
50 dbspaces	.03 X 50	2
15 users	10.28 X 15	154
1 package dbspace	8.10 X 1	8
2 character sets	.25 X 2	1
20 keys	.13 X 20	3
Total number of SYS0001 pages = 418		
Rounded to the next higher multiple of 128 is: 512		

For an Application Development Database

Table 48 on page 351 illustrates the estimate for a medium sized set of catalog tables that might be used in generating a test database to support development of multiple application systems. The number of package dbspaces needed was determined by adding the number of views to the number of packages and dividing the sum by 255. The maximum number of packages that can be defined in a package dbspace is 255. This number could be reduced if the packages are large. The maximum 255 packages may not fit in the allocated pages for the dbspace.

Table 48. Example of Estimating the Catalog Dbspace for an Application Development Database

Example Number of Objects	Number of Pages Calculation	Number of Pages
Reserved	30	30
500 tables	.33 X 500	165
1000 views	.40 X 1000	400
15,000 columns	.10 X 15,000	1,500
50 packages	.50 X 50	25
500 dbspaces	.03 X 500	15
25 users	10.28 X 25	257
5 package dbspaces	8.10 X 5	40
6 character sets	.25 X 6	2
200 keys	.13 X 200	26
Total number of SYS0001 pages = 2461		
Rounded to the next higher multiple of 128 is: 2560		

For a Production Database

Table 49 illustrates the estimate for a large sized set of catalog tables that could be used to support a production database.

Table 49. Example of Estimating the Catalog Dbspace for a Production Database

Example Number of Objects	Number of Pages Calculation	Number of Pages
Reserved	30	30
3000 tables	.33 X 3000	990
5000 views	.40 X 5000	2000
75,000 columns	.10 X 75,000	7500
250 packages	.50 X 250	125
500 dbspaces	.03 X 500	15
50 users	10.28 X 50	514
21 package dbspaces	8.10 X 21	170
6 character sets	.25 X 6	2
1,200 keys	.13 X 1 200	156
Total number of SYS0001 pages = 11,502		
Rounded to the next higher multiple of 128 is: 11,520		

Modifying the SYS0001 Storage Estimating General Formula

Table 50 on page 352 and Table 51 on page 352 assist you if you want to modify any of the assumptions used in deriving the general formula. If you have generated the starter database, you should compare the data in the catalog tables against the assumptions made here. You can do so by issuing UPDATE STATISTICS for each of the catalog tables after you have used the starter database. Queries against SYSTEM.SYSCATALOG give you the statistics for comparison.

Table 50. Your Estimated Stored Lengths of Catalog Rows

Catalog Table	Minimum Length	Maximum Length	Estimated Average Length
SYSACCESS	46	64	
SYSCATALOG	64	384	
SYSCCSIDS	39	39	39
SYSCHARSETS	393	411	
SYSCOLAUTH	46	82	
SYSCOLSTATS	27	123	
SYSCOLUMNS	56	400	
SYSDBSPACES	40	58	
SYSDROP	13	13	13
SYSINDEXES	62	232	
SYSKEYCOLS	55	91	
SYSKEYS	77	113	
SYSOPTIONS	13	303	
SYSPARMS	82	82	
SYSPROGAUTH	46	54	
SYSPSERVERS	11	281	
SYSROUTINES	58	581	
SYSSTRINGS	286	286	286
SYSSYNONYMS	26	62	
SYSTABAUTH	57	101	
SYSUSAGE	36	72	
SYSUSERAUTH	35	35	35
SYSVIEWS	20	292	

Table 51. Your Assumptions of Catalog Bytes or Pages for Each Object

Object	Catalog Entries	Bytes	Bytes for Each Object	Pages for Each Object
Table	1 SYSCATALOG 1 SYSTABAUTH __ SYSINDEXES	___ ___ ___		
View	1 SYSCATALOG 1 SYSVIEWS __ SYSTABAUTH __ SYSUSAGE	___ ___ ___ ___		
Column	1 SYSCOLUMNS	___		
Package	1 SYSPROGAUTH ___ SYSUSAGE	___ ___		
Dbspace (including package dbspaces)	1 SYSDBSPACES	___		

Table 51. Your Assumptions of Catalog Bytes or Pages for Each Object (continued)

Object	Catalog Entries	Bytes	Bytes for Each Object	Pages for Each Object
User	1 SYSUSERAUTH ___ SYSTABAUTH ___ SYSSYNONYMS ____ SYSCOLAUTH	35 ___ ___ ___		
Package dbspaces	255 SYSACCESS	___		
Character Set	1 SYSCHARSETS	___		
Keys	1 SYSKEYS ___ SYSKEYCOLS	___ ___		
Other	3 SYSOPTIONS	___		

Estimating ISQL Dbspace Requirements

An allocation of 1 024 pages should be sufficient for most ISQL users. If you have many users or expect to make extensive use of the ISQL stored queries facility, consider increasing this allocation.

The recommended size (in pages) for the PUBLIC.ISQL dbspace is 1024 or .88 x the number of stored queries, whichever is larger.

Estimating Dbspace Sizes for Routines

The size of ROUTINE tables can vary greatly from user to user and from installation to installation. You can place the ROUTINE tables for all users in the same public dbspace, or you can place the ROUTINE table for each user in that user's private dbspace.

The following formulas are condensed versions of size estimation formulas in the *DB2 Server for VSE & VM Database Administration* manual. They simplify the work required to estimate the size of the dbspace needed to hold routines. The following assumptions have been used in the formulas:
- The PCTFREE value is 15.
- The PCTINDEX value is 33.
- ALLOWANCE was not included in the formula.

The following formula can be used to calculate the average row length for the routines:

```
AVGROWLEN = 23 + average command line length
               + average remark length
```

Note: The average command line length is not the same as the average command length. A command can be entered on multiple command lines. A command line in a routine has a maximum length of 254 characters. A command has a maximum length of 2048 characters. Be sure to use the command line length in your estimate.

The following formula can be used to calculate the number of dbspace pages required for your ROUTINE tables:

```
                  number         average          average
                    of     x     number of   x    number
                  users          routines         of lines
              ________________________________________________

Dbspace pages =    (2074 / AVGROWLEN) (from previous formula)
```

Examples:

Table 52. Example of Estimating the Number of Dbspace Pages for a Routine

Number of Users	**Number of Routines For Each User**	**Number of Lines For Each Routine**	**Row Length**	**Dbspace Pages**
1	20	20	80	16
20	20	20	80	309
50	60	35	75	3797

Estimating Dbspace Size for Stored SQL Statements (Stored Queries)

The following assumptions are used in the formula for calculating the size of the dbspace required for stored SQL statements:

- The PCTFREE value is 15.
- The PCTINDEX value is 33.
- ALLOWANCE is not included in the formula.
- One page was included for one routine named PROFILE. It can contain up to 25 lines with an average row length of 80.

The following formula can be used to calculate the number of dbspace pages needed for stored SQL statements:

```
Dbspace pages = 1 + (.037 x number of statements) +

((Truncate [(avglen + 250) / 250 x 250)]  x number of statements
---------------------------------------------------------------
                  2667
```

When calculating the average length of your stored queries, you must include the FORMAT information for all SELECT statements. The length of the FORMAT information can be calculated by the following formula:

```
Format length = 504 + (number of columns x 44)

                or

                2048, whichever is smaller
```

The following examples show the number of dbspace pages required for each user for the two types of stored SQL statements. The two types are:

- SQL SELECT statements (true stored queries)
- Other SQL statements.

The examples in Table 53 on page 355 and Table 54 on page 355 show the number of dbspace pages required for one user for 10 stored SQL statements. If a user has some of each type of stored SQL statement, you must add the values from each table as needed.

Table 53. Examples — Dbspace Pages for Each User for Stored SELECT Statements

Number of Selects	Number of Columns	Length of Select	Format Length	Adjusted Length[1]	Dbspace Pages for Each User
10	10	70	944	1250	5.88
10	20	70	1384	1500	5.99
10	10	400	944	1500	5.99
10	20	400	1384	2000	7.87
10	40	400	2048	2500	9.38
10	46	2048	2048	4250	16.30

[1] The adjusted length is the stored data length of the stored SQL statements. For more information on adjusted lengths columns, see the *DB2 Server for VSE & VM Database Administration* manual.

Table 54. Examples — Dbspace Pages for Each User for Stored SQL Statements Other than SELECTs

Number of Commands	Average Length	Adjusted Length[1]	Pages for Each User
10	70	250	1.31
10	400	500	2.25
10	999	1 000	4.12
10	1 499	1500	6.00
10	2 048	2 250	8.81

[1]The adjusted length is the stored data length of the stored SQL statements. For more information on adjusted lengths columns, see the *DB2 Server for VSE & VM Database Administration* manual.

Appendix C. Maximum Values

Database Manager Maximum Values

Table 55. Database Manager Maximum Values

Restricted Parameter	**Maximum**
Databases for each system [1]	unlimited
Number of CICS/VSE shareable links	64
Initialization parameters:	
RMTUSERS [2]	65535
DISPBIAS	10
NCUSERS [2]	252
NPACKAGE [2]	32766
NPACKPCT	100
NPAGBUF [2] [3]	400000
NDIRBUF [2] [4]	400000
NLRBU [2]	583333
NLRBS [2]	583333
NCSCANS [2]	655
CHKINTVL [2]	99999999
SLOGCUSH	90
ARCHPCT	99
SOSLEVEL	100

Notes for Table 55:

1. Only one database at a time can be operated in multiple user mode.
2. This is the absolute maximum value. The practical maximum value is less, depending on the values specified for other parameters and system resources (such as the amount of storage available). The maximum number of NCUSERS is limited because the program stack storage for each real agent is obtained below 16 megabytes.
3. These are 4 K pages.
4. These are 512-byte pages.

Database Maximum Values

Table 56. Database Maximum Values

Restricted Parameter	**Maximum**
Number of storage pools	999
Number of dbextents	999
Number of dbspaces	32000
Number of bytes per database [1]	64 gigabytes [3]
Number of pages per database [1]	16,777,088
Number of bytes per dbspace [2]	32 gigabytes [3]
Number of pages per dbspace [2]	8,388,480
Size of the directory	1,048,575 4KB pages
Size of a log [4]	524,287 4KB pages
Size of a dbextent	1,048,575 4KB pages
Size of a storage pool [1]	64 gigabytes [3]

Notes for Table 56:

1. This is the absolute maximum size of the database. The practical maximum is lower.
2. This is the absolute maximum size of a dbspace. The practical maximum is lower.
3. A gigabyte is 2^{30} bytes (1,073,741,824).
4. This is the absolute maximum size allowed, but it is much larger than the database can use. See Table 3 on page 15 for more appropriate estimates.

Appendix D. Updating SYSTEM.SYSSTRINGS

The SYSTEM.SYSSTRINGS catalog table contains information on all the CCSID conversions that this product supports. For each CCSID conversion performed, there must be a corresponding row in this table.

To insert a row, follow these steps:

1. Determine the source and target CCSIDs
2. Determine the conversion type

   The conversion type is based on the **encoding scheme** (EBCDIC or ASCII) of the CCSIDs and whether they are for tagging SBCS, mixed, or graphic data. Note that in any conversion that this product supports, the target CCSID is always EBCDIC. The following are conversion types recognized:
   - "SS" (EBCDIC/ASCII SBCS to EBCDIC SBCS)
   - "SM" (EBCDIC/ASCII SBCS to EBCDIC mixed)
   - "MS" (EBCDIC mixed to EBCDIC SBCS)
   - "MM" (EBCDIC mixed to EBCDIC mixed)
   - "PS" (ASCII mixed to EBCDIC SBCS)
   - "PM" (ASCII mixed to EBCDIC mixed)
   - "GG" (EBCDIC/ASCII graphic to EBCDIC graphic)
3. Determine the error byte

   The error byte is only used for SBCS conversions, and therefore applies to all conversion types except for "GG". Be careful what you set it to: if a character in the source gets converted to the error byte, the conversion is terminated and an error occurs. CCSID conversions either have a NULL error byte, or are set as follows for the detection of DBCS characters in the source when they are not allowed:
   - X'0E' used for
     - "SM" conversions where the source CCSID is EBCDIC
     - "MS" conversions
   - X'3E' used for
     - "SS" conversions where the source CCSID is ASCII
     - "SM" conversions where the source CCSID is ASCII
     - "PS" conversions

   CDRA SBCS conversion tables are modified for use in this type of conversion, so that all DBCS first bytes get mapped to X'3E' instead of the original X'3F'.

   For more information, see step 7, “Customize the SBCS Conversion Table”.
4. Determine the substitution byte

   As with the error byte, the substitution byte is not applicable in "GG" type conversions. Whenever a character in the source gets converted to the substitution byte, warning flags are set in the SQLCA. This byte is set based on whether a given conversion table is created using the **enforced subset match** criterion. (For more on this subject, refer to the *Character Data Representation Architecture Level 1, Registry* manual.) If it is, then this byte is set to X'3F', which is the CDRA-defined **SUB** character for conversions with EBCDIC target CCSIDs.
5. Determine the TRANSPROC

   This field only applies in the cases of "MM", "PM", and "GG" type conversions. It contains the name of the DBCS conversion table that is shipped with this product for use in the conversion. If it contains a value other than any of the

DBCS conversion table names that the database manager recognizes, then the value is treated as the name of a user-defined DBCS conversion exit. (For information on creating a user-defined TRANSPROC exit, see “Coding Your Own TRANSPROC Exit” on page 284.)

6. Create the SBCS conversion table

   An SBCS conversion table may be required except in the case of the "GG" conversion type, where it is not applicable. For a conversion type where SBCS conversion applies, it is possible to specify a NULL SBCS conversion table. For example, you can have a mixed-to-mixed conversion where the SBCS CCSIDs of the source and target mixed CCSIDs are the same, in which case you do not need to perform conversion on the SBCS portion(s) of the mixed source data.

   If you require a non-NULL SBCS conversion table, check the catalog table SYSTEM.SYSSTRINGS to see whether it is already supported. If it is not currently supported, then you have to create the conversion table based on the conversion mapping that you define.

   A user-created SBCS conversion table should be in the same format as those supplied by CDRA: that is, a 256-byte string where the byte at offset *n* (starting at offset 0) corresponds to what codepoint *n* in the source CCSID is converted to. You also have to customize the conversion table for use if its source CCSID is ASCII SBCS or ASCII mixed.

7. Customize the SBCS conversion table

   If your SBCS conversion table is to be used for a conversion with an ASCII SBCS or ASCII mixed source CCSID, you will have to modify it for the proper detection of DBCS first bytes. This requires that you determine the ranges of valid DBCS first byte codepoints for the ASCII SBCS source CCSID; then set the contents of the SBCS conversion table at the offsets that correspond to the DBCS first byte codepoints to:

   a. Error byte

      This is required for the following conversions, where DBCS characters are not allowed in the ASCII source:
      1) "SS" conversions where the source CCSID is ASCII
      2) "SM" conversions where the source CCSID is ASCII
      3) "PS" conversions

      In the case of the CDRA-supplied SBCS conversion tables that are shipped for use in the types of conversion mentioned above, the values contained at the DBCS first byte offsets in the conversion tables have been changed from the original X'3F' to X'3E'. This is also the error byte for these conversions. X'3F' remains the substitution byte for these conversions.

      You should also set the DBCS first byte offsets in your conversion table to a unique character, which will also be your error byte.

   b. X'00'

      This is required for "PM" conversions, where DBCS characters in the ASCII source are allowed and are converted. The database manager considers a byte a DBCS first byte if it gets converted to X'00' using the ASCII-to-EBCDIC SBCS conversion table, and if it is not X'00' itself to begin with.

      In the case of the CDRA-supplied SBCS conversion tables that are shipped for use in "PM" conversions, the values contained at the DBCS first byte offsets in the conversion tables used to be X'3F', and have been changed to X'00'. You **must** therefore also set the characters at the DBCS first byte offsets in the conversion table to X'00', in order for DBCS characters to be recognized in the mixed source.

8. Insert the row into SYSTEM.SYSSTRINGS

   You can create a DBSU job to insert the row into SYSTEM.SYSSTRINGS. For examples, review the ARITPOP MACRO which is supplied with the DB2 Server for VSE code. This macro inserts a row into SYSTEM.SYSSTRINGS for every supported conversion between CCSIDs that are supplied with the database manager.

Appendix E. Defining Your Own Character Set

If you cannot use one of the IBM-supplied sample character sets, you can create your own. (You may be able to use one of the character sets in the *Character Data Representation Architecture Level 1, Registry* manual. The CCSIDs in this manual are registered, and already defined.) To create a character set:

1. Identify all characters in your set and their hexadecimal values. For more information, see “Step 1: Identify All Characters in Your Character Set” on page 364.
2. Classify each character.

   The database manager has 12 character classifications that it uses to identify how the character can be used in SQL statements. You must classify those characters in your set that differ from the ENGLISH character set table. For more information, see “Step 2: Classify the Characters” on page 366.
3. Determine the hexadecimal values to which lowercase characters are to be translated. For example, suppose X'6A' is used for lowercase n-tilde, and X'7B' is used for uppercase N-tilde. The database manager does not automatically know that X'6A' should be converted to X'7B'. You must define that relationship. For more information, see “Step 3: Determine Translation Characters” on page 374.
4. Update the SYSTEM.SYSCHARSETS catalog table.

   Having defined character classifications and translations, you must implement them on the database manager. You can code an INSERT command for the DBS utility to process. During this process, you must also choose a name for the character set (for example, PORTUGUESE). For more information, see “Step 4: Update the SYSTEM.SYSCHARSETS Catalog Table” on page 376.
5. Update the SYSTEM.SYSCCSIDS catalog table.

   When the new character set is implemented, you must add a row to the SYSTEM.SYSCCSIDS catalog table to identify the CCSID values to be associated with the new character set. For more information, see “Step 5: Update the SYSTEM.SYSCCSIDS Catalog Table” on page 376.
6. Update the SYSTEM.SYSSTRINGS catalog table.

   After you specify the CCSID values, you must indicate the conversion table information to allow conversions to and from the new CCSID.
7. Update the CCSID-Related Phases

   Before using the new character set you must run the job control program ARISCNVD. This job copies the information in SYSTEM.SYSCHARSETS, SYSTEM.SYSCCSIDS, and SYSTEM.SYSSTRINGS to three phases, which are used by the VSE application server and application requester. See the *DB2 Server for VSE Program Directory* for more information.

When your character set is loaded into the SYSTEM.SYSCHARSETS catalog table, and the SYSTEM.SYSCCSIDS and SYSTEM.SYSSTRINGS catalog tables have been updated, you can specify the character set by using the CHARNAME initialization parameter.

Each step in defining a character set is discussed below. As an example, a character set (PORTUGUESE) for Brazilian Portuguese is defined.

Step 1: Identify All Characters in Your Character Set

Identify all characters in your character set, and write a matrix like the one shown in the previous figures. For example, Figure 124 shows an example of a character set that might be used to represent Brazilian Portuguese.

		00				01				10				11				Bits 0,1
	Hex 1	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	2,3
Bits 4567		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Hex 0
0000	0					SP	&	–						õ	é		0	
0001	1							/		a	j			A	J		1	
0010	2									b	k	s		B	K	S	2	
0011	3									c	l	t		C	L	T	3	
0100	4									d	m	u		D	M	U	4	
0101	5									e	n	v		E	N	V	5	
0110	6									f	o	w		F	O	W	6	
0111	7									g	p	x		G	P	X	7	
1000	8									h	q	y		H	Q	Y	8	
1001	9								ã	i	r	z		I	R	Z	9	
1010	A					É	$	ç	:				¬					
1011	B					.	Ç	,	Õ				|					
1100	C					<	*	%	Ã									
1101	D					(	)	_	'									
1110	E					+	;	>	=									
1111	F					!	⌢	?	"									

Figure 124. PORTUGUESE Character Set

Figure 125 on page 365 is provided for you to record your own character set.

	4	5	6	7	8	9	A	B	C	D	E	F
0												
1												
2												
3												
4												
5												
6												
7												
8												
9												
A												
B												
C												
D												
E												
F												

Figure 125. Your SBCS Character Set

At this time, you should note the hexadecimal values in your character set that have representations different from those in the ENGLISH character set in Figure 95 on page 236. Recording the differences makes character classification easier. These are the hexadecimal values that have different representations in the PORTUGUESE example:

X'4A'	X'5F'	X'7C'
X'4F'	X'6A'	X'C0'
X'5A'	X'79'	X'D0'
X'5B'	X'7B'	

Step 2: Classify the Characters

When interpreting commands, the database manager must identify which characters are valid, and which are not. To do this, the database manager uses an internal character classification table.

In the table, each of the 256 possible SBCS hexadecimal values are assigned a classification. The database manager uses these classifications to tell whether a character is, for example, a delimiter or a numeric. There are 12 classes. Each hexadecimal value is assigned one of these classes. The only hexadecimal values that you are able to reclassify are those that, in the ENGLISH character set, are classified as 3 or `0`. Values classified as `0` can be reclassified as 3, and values classified as 3 can be reclassified as `0`. No other reclassifications are allowed. The only exception to this rule occurs with certain class 6 characters. See the class 6 description below for details. See Table 57 on page 370 for the ENGLISH character set class. Other character classes are shown only for reference:

Class **Meaning**

0 Unusable for keywords or unquoted identifiers

Any hexadecimal value assigned to this class cannot be used in keywords or unquoted identifiers.

1 Blank

The hexadecimal value assigned to this class is a blank. A blank, in the SQL language, is a delimiter between keywords. The database manager uses X'40' for blanks.

2 Apostrophe

The hexadecimal value assigned to this class is an apostrophe ('). An apostrophe, in the SQL language, is the delimiter for character constants. The database manager uses X'7D' for an apostrophe.

3 Characters other than numerics, uppercase English alphabetics, and underscores that are usable for unquoted identifiers

Numeric, uppercase English alphabetics, and underscores all belong to other classes. In the default ENGLISH character set, the lowercase alphabetics along with $, #, and .* are assigned to this class. In the sample character sets, characters such as n-tilde and o-umlaut are assigned this class.

4 Numerics

Any hexadecimal value assigned to this class is a numeric. The SQL language defines the X'F0' to X'F9' to represent the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. You must not assign class 4 to any other hexadecimal values, nor can you reassign hexadecimal values X'F0' to X'F9' to some other class.

5 Period

Any hexadecimal value assigned to this class is a period. A period, in the SQL language, is the delimiter between a qualifier (such as an owner) and a data object (such as a table). The database manager uses X'4B' for a period.

6 Special characters

Hexadecimal values assigned to this class have special meanings in the SQL language, just as numerics do. You must not assign class 6 to any hexadecimal values other than those listed below. Nor can you reassign the hexadecimal values shown to some other class. The only exceptions are the ones which have a different hexadecimal value depending on the application server default CCSID used. For those hexadecimal values listed which map to a character used in the SQL language for your application server default CCSID, do not reassign these values. For those hexadecimal values listed which do not map to a character used in the SQL language for your application server default CCSID, you can assign them to class 0 or class 3.

For example, X'5A' maps to the exclamation mark (!) for CCSID 37. For CCSID 500, X'5A' maps to the right square bracket (]). For CCSID 37, the hexadecimal value should be class 6. For CCSID 500, the hexadecimal value could be either class 0 or class 3. In the SQL language the following hex values have these meanings:

X'4C'	<
X'4D'	(
X'4E'	+
X'4F'	ǀ (for CCSIDs 37, 284, 285, 290, 420, 424, 833, 836, 838, 1027, 28709)
X'4F'	! (for CCSIDs 273, 277, 278, 280, 297, 500, 870, 871, 875)
X'50'	&
X'5A'	! (for CCSIDs 37, 285, 290, 420, 424, 833, 836, 838, 1027, 28709)
X'5C'	*
X'5D'	)
X'5E'	;
X'5F'	¬ (for CCSIDs 37, 284, 285, 290, 420, 424, 833, 836, 838, 1027, 28709)
X'5F'	^ (for CCSIDs 273, 277, 278, 280, 297, 500, 870, 875)
X'60'	-
X'61'	/
X'69'	^ (for CCSID 838)
X'6A'	ǀ (for CCSIDs 870, 878)
X'6B'	,
X'6C'	%
X'6E'	>
X'6F'	?
X'7A'	:
X'7E'	=
X'B0'	^ (for CCSIDs 37, 290, 424, 833, 836, 1027, 28709)
X'BA'	¬ (for CCSIDs 273, 277, 278, 280, 297, 50 0, 871)
X'BA'	^ (for CCSIDs 284, 285)
X'BB'	ǀ (for CCSIDs 273, 277, 278, 280, 297, 500, 871)
X'BB'	! (for CCSID 284)

X'EC'	^ (for CCSID 871)
X'EF'	¬ (for CCSID 875)

7 Quotation Mark

Any hexadecimal value assigned to this class is a double quotation mark ("). A double quotation mark, in the SQL language, is the delimiter for quoted identifiers. The database manager uses X'7F' for a double quotation mark.

8 Shift-out character

You should not assign any hexadecimal value to this class. When the DBCS option is YES, the database manager assigns this class to X'0E'.

9 Shift-in character

You should not assign any hexadecimal value to this class. When the DBCS option is YES, the database manager assigns this class to X'0F'.

A English Uppercase Alphabetics

This class is restricted to all English uppercase alphabetics (hexadecimal values X'C1' to X'C9', X'D1' to X'D9', and X'E2' to X'E9'). English uppercase alphabetics can be used in unquoted identifiers and keywords. (This is true no matter what SBCS character set is specified.)

B Underscore

Any hexadecimal value assigned to this class is an underscore. An underscore, in the SQL language, can be used in an unquoted identifier except as a starting character. The database manager uses X'6D' for an underscore.

When you have defined a character set, you must classify each hexadecimal value that has a different representation in your character set than it does in the ENGLISH character set.

The database manager always sets the first 64 hexadecimal values (X'00' to X'3F') to class `0`. You can set only the remaining 192 hexadecimal values. Therefore, if any character in your set has a hexadecimal value within X'00' to X'3F', you can use that hexadecimal value only in quoted identifiers.

The only hexadecimal values that the database manager reclassifies in the first 64 are X'0E' and X'0F'. Those hexadecimal values are permanently defined to the database manager as the DBCS shift-out and shift-in characters. When the DBCS option is YES, the database manager reclassifies X'0E' to class 8 and X'0F' to class 9. For more information, see "Using Double-Byte Character Set (DBCS)" on page 243.

Not all SBCS character sets can be classified for use with the database manager, because it reserves certain hexadecimal values. For example, all hexadecimal values that (in the ENGLISH character set) represent uppercase English letters are reserved. The database manager reserves hexadecimal values so it can correctly interpret SQL statements.

Use Table 57 on page 370 to classify your character set. The first column gives the hexadecimal value. The next two columns identify the ENGLISH classification and conversion values for each of those hexadecimal values. (Translation values are discussed in the next step.) The fourth and fifth columns show the classification and conversion values for the PORTUGUESE example. The remaining two columns are for your own character set.

Note: All hexadecimal values are reserved except those that are classified as 0 or 3 in the ENGLISH character set.

Any hexadecimal value that is classified in the ENGLISH character set as 0 or 3 can be reclassified as 3 or 0. Keep in mind that all hexadecimal values that are classified as 0 cannot be used in keywords and unquoted identifiers. Therefore, you would not want to classify as 0 any letter that is within your language's alphabet. You would not be able to use those letters in unquoted identifiers.

The English alphabet consists of the following letters: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and Z. Most likely, the hexadecimal values for letters in your language that are not in the English alphabet are classified as 0 in the ENGLISH character set. You would typically change the classification to 3.

If you must reclassify a hexadecimal value, but the hexadecimal value is reserved, then it is not possible to completely classify the character set. In that situation, it may not be to your advantage to specify an alternative character set. For example, if a character in your alphabet has a hexadecimal value that is assigned class 6 in the ENGLISH character set, you cannot reclassify that hexadecimal value (the only exceptions are the hexadecimal values associated with the characters |, !, ¬ and ^).

The rationale used in classifying the PORTUGUESE character set hexadecimal values that are different from the ENGLISH character set is as follows:

X'4A' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value was reclassified from 0 to 3.

X'4F' The character represented by this hexadecimal value in the PORTUGUESE character set is the exclamation mark. Since this is also a special character in the ENGLISH character set and is already classified as 6, there is no need to reclassify it.

X'5A' This hexadecimal value, which represents a dollar sign ($) in the PORTUGUESE character set, was reclassified from 6 to 0. This was done because the dollar sign is not a special character in the SQL language. If you want to use the dollar sign in unquoted identifiers and keywords, however, you can reclassify it to 3.

X'5B' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value should be classified as 3. It is already classified as a 3 in the ENGLISH classifications, so there is no need to reclassify it.

X'5F' This is another character that is reserved in the SQL language. In the example PORTUGUESE character set, X'5F' represents a caret. Since this is also a special character in the ENGLISH character set and is already classified as 6, there is no need to reclassify it.

X'6A' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value was reclassified from 0 to 3.

X'79' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value was reclassified from 0 to 3.

X'7B' Because the character represented by this hexadecimal value is in the

Portuguese alphabet, the hexadecimal value should be classified as 3. It is already classified as a 3 in the ENGLISH classifications, so there is no need to reclassify it.

X'7C' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value should be classified as 3. It is already classified as a 3 in the ENGLISH classifications, so there is no need to reclassify it.

X'B0' This hexadecimal value, which represents a cent sign (¢) in the PORTUGUESE character set, was reclassified from 6 to 0. This was done because the cent sign is not a special character in the SQL language.

X'BA' The character represented by this hexadecimal value in the PORTUGUESE character set is the NOT sign. Since this is a special character in the SQL language, the value was reclassified from 0 to 6.

X'BB' The character represented by this hexadecimal value in the PORTUGUESE character set is the vertical bar. Since this is a special character in the SQL language, the value was reclassified from 0 to 6.

X'C0' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value was reclassified from 0 to 3.

X'D0' Because the character represented by this hexadecimal value is in the Portuguese alphabet, the hexadecimal value was reclassified from 0 to 3.

Having reclassified the characters, you next need to consider the translation values of those characters.

Table 57. Character Classification and Translation Table

Hex Value	English Class.	English Trans.	Brazilian Class.	Brazilian Trans.	Your Class.	Your Trans.
40	1	40				
41	0	41				
42	0	42				
43	0	43				
44	0	44				
45	0	45				
46	0	46				
47	0	47				
48	0	48				
49	0	49				
4A	0	4A	3			
4B	5	4B				
4C	6	4C				
4D	6	4D				
4E	6	4E				
4F	6	4F				

Table 57. Character Classification and Translation Table (continued)

Hex Value	English Class.	English Trans.	Brazilian Class.	Brazilian Trans.	Your Class.	Your Trans.
50	6	50				
51	0	51				
52	0	52				
53	0	53				
54	0	54				
55	0	55				
56	0	56				
57	0	57				
58	0	58				
59	0	59				
5A	6	5A	0			
5B	3	5B				
5C	6	5C				
5D	6	5D				
5E	6	5E				
5F	6	5F				
60	6	60				
61	6	61				
62	0	62				
63	0	63				
64	0	64				
65	0	65				
66	0	66				
67	0	67				
68	0	68				
69	0	69				
6A	0	6A	3	X'5B'		
6B	6	6B				
6C	6	6C				
6D	B	6D				
6E	6	6E				
6F	6	6F				
70	0	70				
71	0	71				
72	0	72				
73	0	73				
74	0	74				
75	0	75				
76	0	76				
77	0	77				
78	0	78				
79	0	79	3	X'7C'		
7A	6	7A				
7B	3	7B				
7C	3	7C				
7D	2	7D				
7E	6	7E				
7F	7	7F				

Table 57. Character Classification and Translation Table (continued)

Hex Value	English Class.	English Trans.	Brazilian Class.	Brazilian Trans.	Your Class.	Your Trans.
80	0	80				
81	3	C1				
82	3	C2				
83	3	C3				
84	3	C4				
85	3	C5				
86	3	C6				
87	3	C7				
88	3	C8				
89	3	C9				
8A	0	8A				
8B	0	8B				
8C	0	8C				
8D	0	8D				
8E	0	8E				
8F	0	8F				
90	0	90				
91	3	D1				
92	3	D2				
93	3	D3				
94	3	D4				
95	3	D5				
96	3	D6				
97	3	D7				
98	3	D8				
99	3	D9				
9A	0	9A				
9B	0	9B				
9C	0	9C				
9D	0	9D				
9E	0	9E				
9F	0	9F				
A0	0	A0				
A1	0	A1				
A2	3	E2				
A3	3	E3				
A4	3	E4				
A5	3	E5				
A6	3	E6				
A7	3	E7				
A8	3	E8				
A9	3	E9				
AA	0	AA				
AB	0	AB				
AC	0	AC				
AD	0	AD				
AE	0	AE				
AF	0	AF				

Table 57. Character Classification and Translation Table (continued)

Hex Value	English Class.	English Trans.	Brazilian Class.	Brazilian Trans.	Your Class.	Your Trans.
B0	6	B0	0			
B1	0	B1				
B2	0	B2				
B3	0	B3				
B4	0	B4				
B5	0	B5				
B6	0	B6				
B7	0	B7				
B8	0	B8				
B9	0	B9				
BA	0	BA	6			
BB	0	BB	6			
BC	0	BC				
BD	0	BD				
BE	0	BE				
BF	0	BF				
C0	0	C0	3	X'7B'		
C1	A	C1				
C2	A	C2				
C3	A	C3				
C4	A	C4				
C5	A	C5				
C6	A	C6				
C7	A	C7				
C8	A	C8				
C9	A	C9				
CA	0	CA				
CB	0	CB				
CC	0	CC				
CD	0	CD				
CE	0	CE				
CF	0	CF				
D0	0	D0	3	X'4A'		
D1	A	D1				
D2	A	D2				
D3	A	D3				
D4	A	D4				
D5	A	D5				
D6	A	D6				
D7	A	D7				
D8	A	D8				
D9	A	D9				
DA	0	DA				
DB	0	DB				
DC	0	DC				
DD	0	DD				
DE	0	DE				
DF	0	DF				

Table 57. Character Classification and Translation Table (continued)

Hex Value	English Class.	English Trans.	Brazilian Class.	Brazilian Trans.	Your Class.	Your Trans.
E0	0	E0				
E1	0	E1				
E2	A	E2				
E3	A	E3				
E4	A	E4				
E5	A	E5				
E6	A	E6				
E7	A	E7				
E8	A	E8				
E9	A	E9				
EA	0	EA				
EB	0	EB				
EC	0	EC				
ED	0	ED				
EE	0	EE				
EF	0	EF				
F0	4	F0				
F1	4	F1				
F2	4	F2				
F3	4	F3				
F4	4	F4				
F5	4	F5				
F6	4	F6				
F7	4	F7				
F8	4	F8				
F9	4	F9				
FA	0	FA				
FB	0	FB				
FC	0	FC				
FD	0	FD				
FE	0	FE				
FF	0	FF				

Step 3: Determine Translation Characters

When the database manager translates a character string from lowercase to uppercase, it checks the classification of each character in the string. If the character is in class 3, it is translated. If not, the character is not changed.

To translate the character, the database manager consults a translation table. The translation table contains the hexadecimal value to which a particular hexadecimal value is to be translated.

For every hexadecimal value in your set that has a different character representation than in English, you must define a translation value. Refer again to Table 57 on page 370. The following rationale was used to choose the PORTUGUESE translation values:

X'4A' Because the character represented by this hexadecimal value is an uppercase E with an accent, there is no need to translate the hexadecimal value to some other value when the database manager is folding to uppercase. The translation value remains X'4A' (the same as the ENGLISH value).

X'4F' The character represented by this hexadecimal value in Portuguese is the exclamation mark. Because the hexadecimal value should not be changed when the database manager is doing a lowercase to uppercase translation, the translation value should remain X'4F'.

X'5A' This hexadecimal value, which represents a dollar sign ($) in the PORTUGUESE character set, should not change during a translation. The translation value should remain X'5A'.

X'5B' Because this hexadecimal value represents an uppercase C with a cedilla, it does not need to be changed during a translation. The translation value should remain X'5B'.

X'5F' This is another hexadecimal value that does not represent a letter of the alphabet. The hexadecimal value should not change during a translation. The translation value should remain X'5F'.

X'6A' This hexadecimal value represents a lowercase c with a cedilla. During a translation, it should be translated to an uppercase C with a cedilla. To have the database manager do the correct translation, the translation value should be X'5B' (the hexadecimal value for uppercase C with a cedilla).

X'79' This hexadecimal value represents a lowercase a with a tilde. During a translation, it should be translated to an uppercase A with a tilde. The translation value should be X'7C'.

X'7B' Because this hexadecimal value represents an uppercase O with a tilde, it does not need to be changed during a translation. The translation value should remain X'7B'.

X'7C' Because this hexadecimal value represents an uppercase A with a tilde, it does not need to be changed during a translation. The translation value should remain X'7C'.

X'C0' This hexadecimal value represents a lowercase o with a tilde. During a translation, it should be translated to an uppercase O with a tilde. To have the database manager do the correct translation, the translation value should be X'7B' (the hexadecimal value for uppercase O with a tilde).

X'D0' This hexadecimal value represents a lowercase e with an accent. During a translation, it should be translated to an uppercase E with an accent. To have the database manager do the correct translation, the translation value should be X'4A' (the hexadecimal value for uppercase E with an accent).

After determining what the translation values are, verify the following:

1. Hexadecimal values that you have reclassified to class `0` must be translated into identical hexadecimal values. If you reclassify X'A2' from 3 to `0`, you must ensure that the translation value is set to X'A2', not X'E2' (as it is in ENGLISH). In the PORTUGUESE example, this situation did not occur. No hexadecimal values were reclassified from 3 to `0`.
2. Hexadecimal values that you have reclassified to class 3 can be translated into any hexadecimal value having a class of 3 or A. A quick check of the PORTUGUESE-unique translation values show that the hexadecimal values either translate to themselves or to hexadecimal values having class 3. The PORTUGUESE example is still valid.

If your character set fails either of these tests, there is probably an error either in reclassifications or in the translation values chosen.

Step 4: Update the SYSTEM.SYSCHARSETS Catalog Table

After you define translation values for the characters that require them, load the character set into the SYSTEM.SYSCHARSETS catalog table. The easiest way to load a character set is by modifying a copy of the DBS utility control commands that load the sample ENGLISH character set. The A-type member ARISCHAR contains these control commands.

Change your copy of ARISCHAR to reflect the classification and translation values for your character set.

The first value in the INSERT statement is the name of the character set. For 'ENGLISH' substitute the name of your character set. You can specify up to eighteen characters. The value 'PORTUGUESE' was chosen as the name of the example Brazilian Portuguese SBCS character set.

The second value in the INSERT statement contains data for the character classification table. There are 192 character classifications that you can set. You should change only those character classifications in your character set that differ from the ENGLISH classifications. Use the values you have recorded in Table 57 on page 370.

The third value in the INSERT statement contains data for the character translation table. There are 192 character translation values you can set. You should change only those translation values in your character set that differ from the ENGLISH translation values. Use the values you have recorded in Table 57 on page 370. Note that the single quote (X'7D') must be entered twice. A single quote normally delimits the end of a value in an INSERT statement. To use a single quote as part of the data, the single quote must be entered twice.

Step 5: Update the SYSTEM.SYSCCSIDS Catalog Table

You must add a row to the SYSTEM.SYSCCSIDS catalog table to identify the CCSID values to be associated with your new character set. You could issue the following statement to update SYSTEM.SYSCCSIDS for the character set defined for this example:

```
INSERT INTO SYSTEM.SYSCCSIDS (CCSID,SUBTYPE,DBCSID,SBCSID,CHARNAME)
  VALUES (57344,
          'S',
          0,
          0,
          'PORTUGUESE')
```

If you are defining your own CCSID (that is, one that is not obtained from the Character Data Representation Architecture (CDRA)) registry, you must use a value that is within the range of 57 344 to 61 439 (X'E000' to X'EFFF'). Values within this range are reserved for user-defined CCSIDs. Ensure that the value you specify does not already exist: the CCSID column cannot contain duplicate information. Also keep the following in mind:

- If the character set that you are defining uses conversion tables that are provided by the CDRA registry, use the CCSIDs that they indicate.
- The SUBTYPE column identifies the subtype of the CCSID. In this example, the value is 'S' for SBCS.
- The SBCSID column and the DBCSID column specify the SBCS and the DBCS components for a mixed CCSID. Because the CCSID in this example is SBCS, the value for both of these columns is `0`.

- The value that you specify for the CHARNAME column must be the same as the value that you specified in the NAME column of the SYSTEM.SYSCHARSETS catalog table.

For examples of statements that insert rows into the SYSTEM.SYSCCSIDS catalog table, review the ARITPOP MACRO that is supplied with the database manager.

Step 6: Update the SYSTEM.SYSSTRINGS Catalog Table

The SYSTEM.SYSSTRINGS catalog table identifies the tables that will be used for conversion between specific pairs of CCSIDs. Conversion tables for CDRA-supplied CCSIDs are provided by the CDRA registry. For more information on CDRA conversion tables, see the *Character Data Representation Architecture Level 1, Registry* manual. After you create your CCSID, you must determine the conversion table information for SYSTEM.SYSSTRINGS. You must add a row to SYSTEM.SYSSTRINGS for each conversion that you want to support both to and from the new CCSID. For a detailed description to update SYSTEM.SYSSTRINGS, see Appendix D, “Updating SYSTEM.SYSSTRINGS,” on page 359.

Suppose you added CCSID 57344 and you want to support the following conversions:
- CCSID 37 to CCSID 57344
- CCSID 57344 to CCSID 37
- CCSID 57344 to CCSID 28709.

You must add three rows to the SYSTEM.SYSSTRINGS catalog table. To specify any of these conversions in SYSTEM.SYSSTRINGS, you would use an INSERT statement to insert the necessary information into the following columns of the catalog table:
- INCCSID, which specifies the CCSID of the input character.
- OUTCCSID, which specifies the CCSID to which the conversion is done.
- TRANSTYPE, which identifies the type of conversion to be done (for example, 'SS' for SBCS to SBCS).
- ERRORBYTE, which identifies characters that have no representation in the target code page. If a character to be converted maps to a code point containing this byte an error occurs.
- SUBBYTE, which identifies characters that have no representation in the target code page. If a character to be converted maps to a code point identified by this byte, a warning is issued.
- TRANSPROC, which identifies the conversion procedures that are used for conversion between CCSIDs. The procedures are used either for converting between DBCS CCSIDs, or for converting the DBCS components of mixed CCSIDs. The TRANSPROC value is blank if a DBCS conversion procedure is not applied. For more information, see “Coding Your Own TRANSPROC Exit” on page 284.
- TRANSTAB1, which represents the first 64 bytes of the conversion table.
- TRANSTAB2, which represents the last 192 bytes of the conversion table.

The conversion table maps the hexadecimal representation of each character in the source CCSID to the hexadecimal representation of each character in the target CCSID. For example, in CCSID 37 an exclamation mark (!) is represented by X'5A'. The hexadecimal representation for the exclamation mark in CCSID 281 is X'4F'. The hexadecimal value of the character at offset 90 in the conversion table where

INCCSID=37 and OUTCCSID=281 would be X'4F'. Remember, when counting offsets, the *first* offset is *zero*. Therefore, the byte at offset 90 is actually the 91st.

For a detailed description to update SYSTEM.SYSSTRINGS, see Appendix D, "Updating SYSTEM.SYSSTRINGS," on page 359.

When you have completed steps 1 through 6, you can start the application server using the newly defined character set. If the database manager detects an error in the character set, it uses the value of CHARNAME that was used the last time the application server was started.

Step 7: Update the CCSID-Related Phases

After you have updated the SYSTEM.SYSCHARSETS, SYSTEM.SYSCCSIDS and SYSTEM.SYSSTRINGS catalog tables, run the job control program ARISCNVD to load the CCSID information from the catalog tables into three phase files (ARISCCSD, ARISSTRD, and ARISSCRD). The application server and the application requester use these files to enable the use of the new character set. For more information on the control program ARISCNVD, refer to the *DB2 Server for VSE Program Directory*.

When you have completed steps 1 through 7, you can start the application server using the newly defined character set. If the database manager detects an error in the character set, it uses the value of CHARNAME that was used the last time the application server was started.

Appendix F. Macro List

The macros identified in this appendix are provided as programming interfaces for customers by the DB2 Server for VSE database management system.

Attention: Do not use as programming interfaces any DB2 Server for VSE macros other than those identified in this chapter.

Macro list

The database manager provides the following General-use programming interface macros:
- ARIRCAN
- ARIBFPPB.

Appendix G. Service and Maintenance Utilities

This appendix describes the following utilities:
- "SQLDBDEF EXEC," below

SQLDBDEF EXEC

The SQLDBDEF utility extracts the definition of database objects from a DB2 Server for VSE & VM database, and generates a DBSU job that can be used to create the same objects on another DB2 database. The SQL statements in the DBSU job can be grouped in one of two ways:
- By dbspace
- By object type

Note that the SQLDBDEF utility does not generate SQL statements to recreate system tables and dbspaces. However, it does generate SQL statements to recreate user-created views, synonyms, and indexes on system tables.

When the objects have been created on the target platform, the load utilities of the target database can be used to load the data. Packages can be unloaded from the source database and reloaded to the target database so that existing client applications can continue to be used.

Before running the SQLDBDEF utility, you must load the associated packages and link-edit the utility. The job ARISRLDF is provided to load the packages. The linkbook ARISLKRF is provided to link-edit the SQLDBDEF utility.

Authorization

To run the SQLDBDEF EXEC, you must have DBA authority.

Syntax

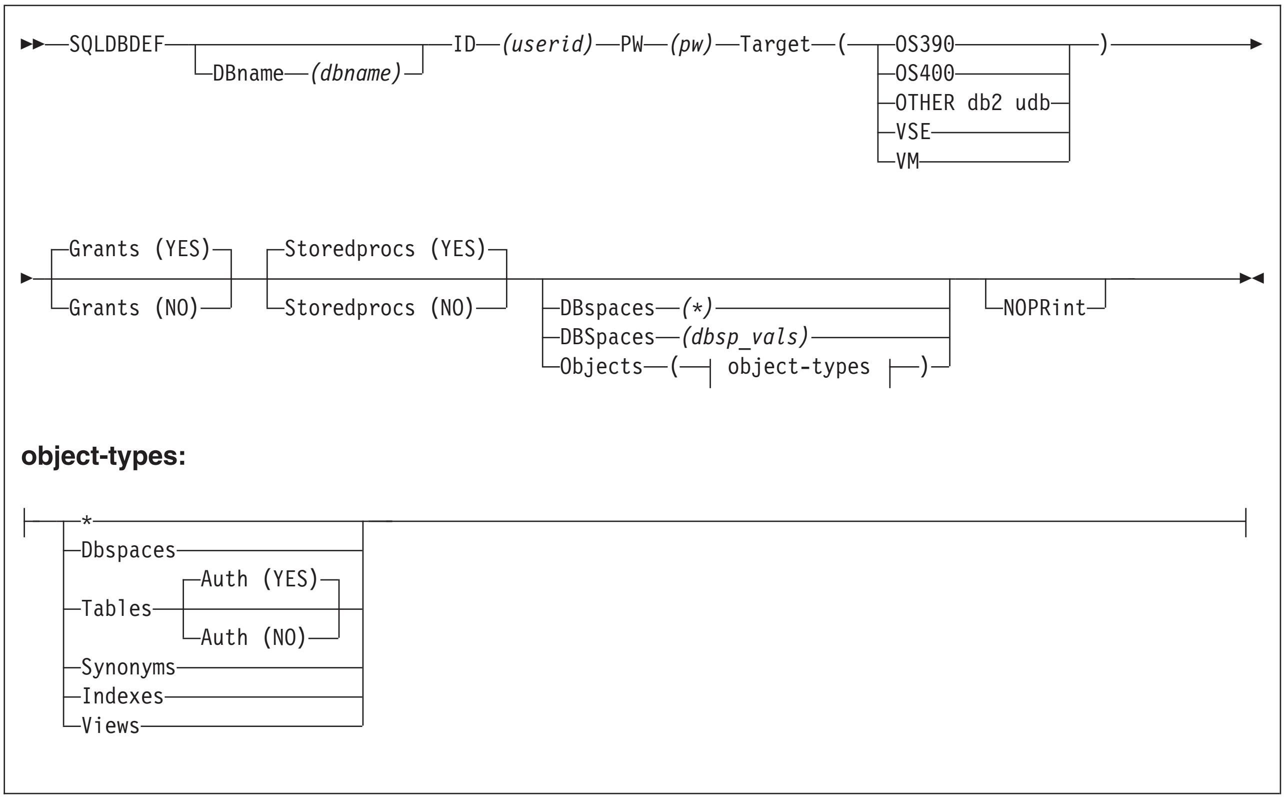


Figure 126. SQLDBDEF Utility

The job control member ARIS215D is provided to invoke the SQLDBDEF utility. The following is a sample invocation:

```
// JOB SQLDBDFJ
// LIBDEF *,SEARCH=(PRD2.DB2730)
// EXEC REXX=SQLDBDEF PARM='DB(DB2VSE) ID(userid) PW(password)'
/*
/&
```

Description

The parameters for the SQLDBDEF utility are as follows:

DBname (*dbname*)
: *dbname* is the DB2 Server for VSE & VM database from which SQL object definitions will be extracted. If the DB parameter is not specified, the database currently defined as the default in the DBname directory will be used.

ID (*userid*)
: *userid* is the connect ID to be used when running the SQLDBDEF utility. The user ID must have DBA authority in order to load the package.

  The ID parameter is required in VSE.

PW (*pw*)
: *pw* is the connect password to be used when running the SQLDBDEF utility. Like the ID parameter, the PW parameter is required in VSE.

Target (OS390), Target (OS400), Target (OTHER db2 udb), Target (VSE), Target (VM)
: This parameter indicates the platform on which the target database runs.

OS390 indicates that the output DBSU job will be run on a DB2 database on MVS, OS/390, or z/OS. **OS400**, **VSE**, and **VM** indicate that the output DBSU job will be run on a DB2 database on OS/400, VSE, or VM respectively. **OTHER db2 udb** indicates that the output DBSU job will be run on a DB2 database on an Intel™ or UNIX platform, for example, Windows, AIX, or Linux.

Grants (YES) Grants (NO)
The Grants parameter determines whether GRANT statements for user authorities should be generated. Note that for GRANT CONNECT statements, the passwords in effect on the source system will not be displayed in the output DBSU job. Instead, the string "<PW>" will be displayed. The default is YES, which causes the utility to generate the GRANT statements.

Storedprocs (YES), Storedprocs (NO)
The Storedprocs parameter determines whether CREATE PROCEDURE statements and (if the target platform is VM or VSE) CREATE PSERVER statements are to be generated. The default is YES, which causes the utility to generate these CREATE statements.

DBSpaces (*), DBSpaces (*dbsp_val*)
This parameter indicates that the SQLDBDEF utility is to generate SQL statements to recreate all objects in one dbspace or in all dbspaces. If this parameter is specified, the utility extracts dbspace, table, view, synonym, and index definitions, as well as table and column privileges, from the system catalog.

If "*" is specified, the output DBSU job contains SQL statements to recreate all objects in all dbspaces, and the statements are grouped by dbspace.

If *dbsp_val* is specified, the output DBSU job contains SQL statements to recreate the objects in that dbspace only. *Dbsp_val* can be a dbspace number or a qualified or unqualified dbspace name. If an unqualified dbspace name is provided, SQLDBDEF looks first for a private dbspace with the specified name that is owned by the connected user. If no private dbspace is found, SQLDBDEF looks for a public dbspace with the specified name.

Note that either the DBSpaces parameter or the Objects parameter can be specified, but not both. If neither is specified, the SQLDBDEF utility does not generate SQL statements to create dbspaces, tables, synonyms, indexes, or views.

Objects (*), Objects (Dbspaces), Objects (Tables), Objects (Synonyms), Objects (Indexes), Objects (Views)
This parameter indicates that the SQLDBDEF utility is to generate SQL statements to recreate one particular type of object, or all objects.

If "*" is specified, the output DBSU job contains SQL statements to recreate all object types, and the statements are grouped by object type.

If a specific object type is specified, the output DBSU job contains SQL statements to recreate objects of that type only.

Note that either the Objects parameter or the DBSpaces parameter can be specified, but not both. If neither is specified, the SQLDBDEF utility does not generate SQL statements to create dbspaces, tables, synonyms, indexes, or views.

Auth (YES), Auth (NO)
If you are generating SQL statements to create all table objects, you can direct the SQLDBDEF utility to generate SQL statements to GRANT existing table

and column privileges as well. The Auth parameter determines whether GRANT statements for table and column privileges are generated. The default is YES.

NOPRint
Specifying NOPRint indicates that no listing file is to be generated. If NOPRint is not specified, a listing file is sent to SYSLST.

The utility generates a DBSU job that can be run to create the database objects (or equivalent objects) on the target database. In order to do this, it must map platform specific SQL statements to their equivalents on the target platform. For example, if the target database is DB2 UDB for OS/390, ACQUIRE DBSPACE statements are mapped to CREATE TABLESPACE.

For SQL clauses that are the same for both the source and target database, the value that was specified for the source database will be used. SQL clauses that can be specified on the target database but are not applicable on the source database do not appear in the generated SQL statement. SQL clauses that can be specified on the source database but are not applicable on the target database do not appear in the generated SQL statement.

The listing goes to SYSLST, and the DBSU job goes to SYSPCH.

Customize the output DBSU job as follows:
1. Ensure that the CONNECT statement specifies the userid and password of a user who has DBA, DBADM, or SYSADM authority (depending on the target platform).
2. Provide valid passwords on the GRANT statements, if applicable.
3. Ensure that the user authorities being granted are appropriate for your installation.
4. Ensure that the CREATE TABLESPACE and ACQUIRE DBSPACE statements are appropriate for your installation.

It is recommended that you review the entire DBSU job prior to running it, to verify that the objects it will create are appropriate for your installation.

Appendix H. DRDA Considerations

Users who are planning to design applications that
- run on non-VSE platforms and use the Distributed Relational Database Architecture (DRDA) protocol to connect to DB2 Server for VSE & VM servers, or
- run on VSE/ESA and use the Distributed Relational Database Architecture (DRDA) protocol to connect to servers other than DB2 Server for VSE & VM

need to be aware that DB2 Server for VSE & VM's support of SQL does not exactly match the IBM SQL standard[6] or the SQL Entry Level standard. [7] This appendix attempts to provide some guidance in discrepancies to these standards.

Omissions from the Standards

For a list of where DB2 Server for VSE & VM does not support the IBM SQL or SQL92 entry level standards, please consult the *DB2 Server for VSE & VM SQL Reference* manual.

Extensions to the Standards

1. There is no support for modifiable packages created by using extended dynamic statements. If you request such support by specifying the MODIFY option on the CREATE PACKAGE statement, the system will override this option with NOMODIFY.
2. Nonmodifiable packages created by using extended dynamic statements are supported with the following restrictions:
    a. There is no support for the positioned UPDATE and positioned DELETE statements.
    b. If the Basic Extended PREPARE form of the extended PREPARE statement prepares a statement that contains parameter markers, the USING DESCRIPTOR clause must be used to identify an input SQLDA structure.
    c. There is no support for the Single Row Extended PREPARE form of the extended PREPARE statement.
    d. There is no support for the NODESCRIBE option of the CREATE PACKAGE statement. If specified, it will be ignored.
    e. There is no support for "USER" in the ISOLATION option of the CREATE PACKAGE statement. The system will override USER with CS.
    f. There is no support for "LOCAL" in the DATE or TIME option of the CREATE PACKAGE statement. If specified, SQLCODE -168 (SQLSTATE 42615) will be generated, indicating an incorrect parameter.
    g. DB2 Server for VSE & VM servers do not support cursors declared with the "WITH HOLD" clause. However, applications may use the "WITH HOLD" clause against other DRDA servers if they support it, except when extended dynamic statements are involved.

6. IBM SQL is a superset of the SQL92 Entry Level standard

7. Entry Level of the International Organization for Standardization (ISO) 9075-1992 Database Language SQL specification

3. There is no support for the semantics checking of the Flagger, but the syntax checking of static SQL against the SAA and SQL-89 standards will still be carried out.

DB2 Server for VSE Facility Restrictions

1. There is no support for “USER” in the CBND parameter ISOLATION. When specified, the system will override USER with CS.
2. There is no support for “LOCAL” in the CBND parameters DATE and TIME. When specified, the remote DRDA application server will return SQLCODE -168 (SQLSTATE 42615).
3. In a private flow environment, there is no support for the blocking of PUTs. However, the PUT operation will still be supported one row at a time as unblocked inserts. In a DRDA flow environment with the DB2 Server for VSE Batch AR, no blocking is provided on a PUT. Each PUT will result in the execution of a single INSERT. Therefore, if large amounts of data are loaded into a remote server, it may be better to transfer the data through some other means to the target site, and then use the local utility to load the data into the database.
4. The following ISQL commands are not supported when ISQL is connected to a remote DRDA application server, because they request functions or depend on the DB2 Server for VSE system catalogs:
   - SET ISOLATION
   - COUNTER
   - SHOW.

   When ISQL is connected to a remote application server and a long-running SQL statement is processed. ISQL will NOT display message ARI7044I to allow the user to enter ISQL CANCEL to cancel the long-running SQL statement. The user cancel exit is not supported on DRDA connections.
5. The DRDA Online Resource Adapter does not provide the user cancel exit to allow online DRDA applications to perform a CANCEL function. For example, a long-running SQL statement cannot be cancelled on a remote connection.
6. The DRDA Online Resource Adapter does not call ARIUXIT when a CICS user tries to connect to a remote DRDA application server either implicitly or explicitly and the accounting exit is link-edited in AMODE 24.

   If the accounting exit is link-edited in AMODE 24 and the CICS user tries to connect to a remote DRDA application server, SQLCODE=-947 will be generated, indicating ARIUXIT in AMODE 24 is not supported.
7. If accounting data is sent from a DRDA application requester to a DB2 for VSE & VM server, only the first 16 bytes of user-defined data [8] is captured by the server and put into accounting records.
8. The current CICS/VSE implementation provides no facility for establishing APPC conversations that use a security level of PGM (for example, a user ID and password are required to allocate the conversation). CICS transactions can only establish SECURITY=SAME conversations to remote APPC partners.

   Due to this limitation, DB2 Server for VSE online application requester may use the "*userid* IDENTIFICATION BY *password*" clause on the CONNECT statement when connecting to a remote application server. In this case, the VSE online application requester performs DRDA security checking as described in the *Distributed Relational Database Architecture Reference* manual.

8. For example, from DDCS for OS/2 user-defined data can be set by the DFT_ACCOUNT_STR configuration parameter.

This check will be performed during handshaking after having established an APPC conversation with a DRDA server that supports security handshaking.

9. A user must sign on to CICS first before executing a transaction that accesses a remote DRDA application server. If a user failed to sign on to CICS and executes a transaction to access a remote DRDA application server, the connect will fail because CICS passes a null user ID to the remote system, which is invalid. DRDA does not support security NONE.
10. The following DBSU commands are not supported when using the DRDA protocol, because they request functions specific to DB2 Server for VSE:
    - UNLOAD DBSPACE
    - UNLOAD TABLE
    - UNLOAD PACKAGE
    - RELOAD DBSPACE
    - RELOAD TABLE
    - SET ISOLATION
    - SET UPDATE STATISTICS
    - REBIND PACKAGE
    - REORGANIZE INDEX

Appendix I. Incompatibilities Between Releases

This appendix identifies the incompatibilities that exist between each release of the product and the previous release, going back to Version 1 Release 3.5. There is a separate section in the appendix for each release.

Note on Skipping Releases: If your migration plans call for skipping one or more releases (for example, migrating directly from V2R2 to V3R4), you will still be affected by the incompatibilities introduced by the releases that you are skipping.

Within each section, the incompatibility items are grouped into the following categories:
- SQL and Data
- Application Programming
- System Environment

Definition of an Incompatibility

For the purpose of this appendix, an "incompatibility" is defined to be a part of the product that works differently than it did in the previous release, in such a way that if used in an existing application, it will produce a different result, necessitate a change to the application, or reduce performance. In this definition, "application" can apply to a broad range of things (singly or in combination), such as:
- Application program code
- Specifications for preprocessing application programs
- Interactive SQL queries
- ISQL functions
- DBS Utility functions
- Miscellaneous tools in your operating environment.

This appendix does not describe incompatibilities where certain operations in the current release are less likely to generate an error condition than they did in the previous release, as those changes will only have a positive impact on your applications. (For example, the SUM and AVG column functions no longer overflow as easily because they now use a larger accumulator, and a change to the use of the equal (=) compare predicate with a negative indicator variable now evaluates to UNKNOWN rather than generating an error condition.)

Impact on Existing Applications

Read the appropriate section of this appendix carefully to determine what changes you will need to make to your applications when migrating from one release to the next. You may also want to review the chapter in the *DB2 Server for VSE System Administration* manual on migration considerations which discusses some of these incompatibilities in more detail, plus other considerations for each release-to-release migration.

This appendix excludes the numerous changes and enhancements for which no impact on existing applications is anticipated. These are listed in the *Summary of Changes* section (included with each manual) of the appropriate release of the library. Review that section to see where you could make changes to your existing applications in order to take advantage of some of these enhancements.

V2R1 and V1R3.5 Incompatibilities

SQL and Data

1. Evaluation of HAVING and SELECT Clauses

   Prior to V2R1, the HAVING clause was evaluated ***after*** the SELECT clause. This caused a statement such as the following to fail on a zero divide and generate SQLCODE -802, if a zero part number was encountered:

   ```
   SELECT 200/PARTNO FROM T1
   GROUP BY PARTNO HAVING PARTNO > 0
   ```

   In V2R1, the HAVING clause is evaluated ***before*** the SELECT clause. This means your applications now have the potential of producing different results. In the above example, if a zero part number is encountered, the query does not fail and SQLCODE -802 is not generated.
2. Null Values as a Grouping Criterion

   Prior to V2R1, if any row had a null value in one of the columns referenced in a GROUP BY clause, each such row was treated as a separate group.

   In V2R1, null values are considered identical for purposes of grouping.

   This means that your existing applications may generate fewer rows in the result table than they did in previous releases, since multiple null-value-groups are now consolidated into one group. Any derived column function values will reflect this consolidation (for example, SUM(BONUS)).
3. Negative Decimal Zero Support

   Prior to V2R1, the system recognized negative decimal zero as a valid value. However, it did not evaluate positive and negative decimal zero values as equivalent.

   In V2R2, any negative decimal zeros found in SQL statements are converted to positive decimal zeros before execution. This means that inserting, updating, or deriving negative decimal zeros, or using them in a comparison, is no longer possible. A utility called SQLZERO is provided which converts all negative decimal zeros in the database to positive decimal zeros.

   For a detailed discussion of this topic, see “Elimination of Negative Decimal Zero” in the chapter which discusses migrating from V1R3.5 in the *System Planning and Administration* manual, V2R1 or later.
4. Insertion of Invalid Decimal Values

   Prior to V2R1, it was possible to insert invalid decimal data into the database during DATALOAD by specifying string values that were invalid for DECIMAL columns. For example, X'0000' has no sign value.

   In V2R1, this is no longer allowed. Doing so will generate SQLCODE -424.

Application Programming

5. Use of ORDER BY Clause with SELECT INTO

   Prior to V2R1, the SELECT INTO statement was allowed to contain an ORDER BY clause.

   In V2R1, this is no longer allowed. Doing so will generate SQLCODE -524.
6. Scope of Prepared Statements

   Prior to V2R1, a prepared statement could sometimes, but not always, be referenced in subsequent logical units of work (LUWs).

In V2R1, this inconsistency is removed. A prepared statement may now only be referenced within the same LUW in which it was prepared.

If your applications contain code that references prepared statements across LUWs, they will have to be restructured accordingly.

7. SQLCODE Returned After a Format 2 INSERT

   Prior to V2R1, when a format 2 INSERT (known as "INSERT via subselect" in V2R2 and later releases) returned an empty answer set for insertion, SQLCODE +0 was generated.

   In V2R1, SQLCODE +100 is generated instead.

8. Preprocessor Errors Converted to Warnings

   Prior to V2R1, a certain set of conditions generated errors during preprocessing.

   In V2R1, these conditions now generate warnings, although the associated SQLCODEs are still negative (starting with V3R1, the codes are presented as positive numbers). These conditions and their corresponding SQLCODEs are shown in the table below.

SQLCODE	DESCRIPTION
-134	IMPROPER USE OF THE LONG FIELD COLUMN column.
-135	THE INPUT FOR A LONG FIELD COLUMN IN AN INSERT OR UPDATE MUST BE FROM A HOST VARIABLE OR THE KEYWORD NULL.
-150	THE VIEW CANNOT BE USED TO MODIFY DATA SINCE IT IS BASED ON MORE THAN ONE TABLE.
-151	A COLUMN OF A VIEW CANNOT BE UPDATED SINCE IT IS DERIVED FROM AN EXPRESSION.
-152	A COLUMN OF A VIEW CANNOT BE USED IN A WHERE-CLAUSE SINCE IT IS DERIVED FROM A COLUMN FUNCTION.
-154	VIEW LIMITATIONS DO NOT ALLOW THE USE OF THE FOLLOWING OPERATION: operation
-155	YOU CANNOT PERFORM A JOIN ON A VIEW CONTAINING A GROUP-BY CLAUSE OR A DISTINCT KEYWORD.
-156	RESTRICTIONS APPLY WHEN SELECTING FROM A VIEW CREATED WITH THE DISTINCT OR GROUP BY KEYWORD.
-202	COLUMN column WAS NOT FOUND IN ANY TABLE REFERENCED BY THE COMMAND.
-205	COLUMN column WAS NOT FOUND IN TABLE creator.table.
-401	INCOMPATIBLE DATA TYPES FOUND IN AN EXPRESSION OR COMPARE OPERATION.
-404	A CHARACTER STRING SPECIFIED IN AN INSERT OR UPDATE IS TOO LARGE FOR THE TARGET COLUMN.
-405	THE NUMERIC VALUE, value, IS NOT WITHIN THE RANGE OF THE DATA TYPE.
-407	AN UPDATE OR INSERT OF A NULL VALUE FOR A COLUMN DEFINED AS NOT NULL IS NOT ALLOWED.
-408	AN UPDATE OR INSERT OF A DATA VALUE IS INCOMPATIBLE WITH THE DATA TYPE OF THE ASSOCIATED TARGET COLUMN.
-414	LIKE WAS USED FOR A NUMERIC OR DATE/TIME COLUMN TYPE. IT MUST ONLY BE USED WITH CHAR OR VARCHAR TYPE COLUMNS.
-415	THE DATA TYPES OF CORRESPONDING ITEMS IN THE SELECT-CLAUSES CONNECTED BY A UNION ARE NOT IDENTICAL.

SQLCODE	DESCRIPTION
-416	YOU CANNOT SPECIFY A LONG FIELD COLUMN IN THE SELECT-CLAUSE OF A UNION.
-419	THE PRECISION OF THE NUMERATOR AND/OR THE SCALE OF THE DENOMINATOR ARE TOO LARGE FOR DECIMAL DIVISION.
-421	A HEXADECIMAL LITERAL WITH AN ODD LENGTH MAY NOT BE USED WITH A DBCS COLUMN IN A PREDICATE.

V2R2 and V2R1 Incompatibilities

SQL and Data

1. Leading and Trailing zeros in Decimal Constants

   Prior to V2R2, leading and trailing zeros of decimal constants were removed by the system when calculating their scale and precision.

   In V2R2, if the precision of a decimal constant is greater than 15, leading zeros are removed to bring the precision down to 15. Trailing zeros are not removed.

   If your current applications provide output from the result table without any intervening formatting, this change has the potential of altering that output. If formatting is involved, you may have to change the formatting logic to obtain the same output.

   Similarly, input to the database by means of INSERT or UPDATE may be affected, if a decimal constant is involved.

2. Use of Host Variables with UNION

   Prior to V2R2, two select-lists could be successfully UNION'ed even when they contained corresponding items that were host variables of different data types and different lengths. The statement below is an example of this, where host variables :hw and :fw are halfword fixed binary (15) and fullword fixed binary (31), respectively.

   ```
   SELECT :hw FROM T1
   UNION
   SELECT :fw FROM T1
   ```

   In V2R2, the above statement is no longer allowed. Issuing it will generate SQLCODE -415.

   Note: In V3R1, some restrictions on the use of data types within a UNION are removed, including the above incompatibility.

   Application Programming

3. Atomic Operations Against the Database

   Prior to V2R2, many types of operational errors (that is, SQL statement errors) against the database caused the system to roll back the entire current logical unit of work (LUW), leaving the application with no control over the status of the LUW.

   In V2R2, all operations against the database are now atomic. That is, within an LUW, each operation can succeed or fail separately, with no effect on other operations, provided they do not depend on it. If an operation fails, the application is free to either continue working on the same LUW, or commit the changes made so far, or roll back the LUW. Some system errors, such as deadlocks, still require the entire LUW to be rolled back by the system. Also, atomic operation is not supported for:

   - Operations on data located in nonrecoverable storage pools

- Operations on data when running without a log (LOGMODE=N).

As a result of this change, you may want to extend the logic of your LUW processing in your applications.

Note: The next incompatibility item contains a special case of atomic operation.

4. Multiple Row Changes Within an Atomic Operation

   Prior to V2R2, if an error occurred during a single operation involving multiple row changes to the database, the database was potentially left in an inconsistent state. (This was one of those operational errors that was not rolled back by the system.) Some of the rows were processed; the rest were not. The only practical way to avoid this inconsistency was to have the application roll back the entire current LUW.

   There was one exception to this: in the case of a data definition statement, such as CREATE TABLE, the system itself rolled back the LUW to avoid a partial definition of a table in the catalog. The application had no control over the status of the LUW.

   In V2R2, with atomic operation in place, the system automatically undoes that portion of the multiple row operation that was processed prior to the error. This eliminates the potential of an inconsistent database resulting from such an operation, and leaves the application free to control the current LUW as it sees fit.

   See “Detailed Notes on V2R2-V2R1 Incompatibilities” on page 394 for an example.

5. Four-Byte Floating-point Data

   Prior to V2R2, all floating-point data had to be eight bytes.

   In V2R2, it can be four bytes.

   This leads to a potential problem in V2R2 for programs that allocate eight bytes when using DESCRIBE on a FLOAT column. When using DESCRIBE, applications should allocate storage based on the SQLLEN of a column (as given in the SQLDA), not the SQLTYPE.

6. Arithmetic and Conversion Errors

   Prior to V2R2, an arithmetic or conversion error terminated processing of the statement and generated SQLCODE -802.

   In V2R2, these types of errors are tolerated when they involve a host variable that has an indicator variable. In such cases, processing of the SQL statement continues; SQLCODE +802 is generated; a -2 is placed in the indicator variable; and the associated database variable remains unchanged.

   If your application is checking for these errors, this could impact its logic. The types of errors that can now be tolerated are:
   - Fixed point overflow
   - Decimal overflow
   - Exponent overflow
   - Exponent underflow
   - Divide exception.

   For more detail, see the *Messages and Codes* manual, V2R2 or later, for SQLCODEs +802 and -802.

7. GRANT Authority for PUBLIC

   Prior to V2R2, “WITH GRANT OPTION” in a GRANT statement passed GRANT authority to the user receiving the privilege in question, even when the user was PUBLIC.

In V2R2, when “PUBLIC” and “WITH GRANT OPTION” are used together, the privilege is granted to PUBLIC, but without GRANT authority. In such cases, a warning is given to that effect.

This can impact your current authorization of views or programs, since these objects, which previously could have been grantable (for example, a value of 'G' recorded for a program in catalog table SYSPROGAUTH), will no longer be so (a value of 'Y' now in SYSPROGAUTH) if they depend on PUBLIC access to an object.

For example, if a program contains a static SELECT statement involving table T1, and the owner of the program is dependent on PUBLIC access to T1, then 'Y' is the highest authorization value attainable for that statement — and therefore for the program. This means that the owner is still able to run the program, but not to grant the RUN privilege on it to others. This, in turn, means that when this program is preprocessed under V2R2, users who previously may have had authority to run it (by virtue of receiving RUN authority from the owner) will no longer have that authority.

System Environment

8. Change to Message Numbers

   Prior to V2R2, the ARI message numbers were three digits long and were followed by an action indicator. This identification formed a header for each line of the message text, as illustrated below:

   ```
   ARI297A  RESPONSE TO ARCHIVE PROMPT
   ARI297A  IS NOT VALID.
   ```

   In V2R2, these message numbers are expanded to four digits to accommodate future expansion of the system. Message numbers existing in the earlier releases now contain a high-order zero. Also, the message header is now only used on the first line of the message. The above example becomes:

   ```
   ARI0297A  RESPONSE TO ARCHIVE PROMPT
             IS NOT VALID.
   ```

   This could impact any automated operating system facility that you may be using (for example, the VM Programmable Operator) to scan the message number and text.

Detailed Notes on V2R2-V2R1 Incompatibilities

1. Multiple Row Changes Within an Atomic Operation

   In the following example, the operations are contained in one LUW. The second operation involves multiple row changes to the database.

   ```
   DELETE FROM SUPPLIER WHERE SUPPNO = 64
   UPDATE INVENTORY SET PARTNO = PARTNO + 1
   INSERT INTO QUOTATIONS VALUES (64, 221, .25, 5, 100)
   ```

   The DELETE statement removes a supplier from the SUPPLIER table. The UPDATE statement changes the first two rows of the INVENTORY table, but fails on the third row because the operation would create a duplicate primary key value.[9]

   Prior to V2R2, the system would have left the new values in the first two rows of INVENTORY, with the rest of the table unchanged. To avoid this undesirable inconsistency, the application would have had to contain logic to recognize this error and roll back the entire LUW, thus undoing the DELETE.

9. In V3R2 this error will not occur, because the enforcement of uniqueness is done after all the rows are updated.

In V2R2, when this error occurs, the system undoes the UPDATE statement by reversing the changes made to the first two rows. Because neither the DELETE nor the INSERT depends on the success of the UPDATE (these operations are atomic), the application has the following options open to it:
- Proceed and perform the INSERT, or
- Commit the successful DELETE, or
- Roll back the LUW to undo the DELETE.

V3R1 and V2R2 Incompatibilities

SQL and Data

1. Table Designation Rules

   Prior to V3R1, the following set of ANS/ISO SQL rules for table designation in FROM clauses were not fully enforced:

   - Duplicate table or view names in a FROM clause must all have a correlation name assigned to them.
   - Correlation names in a FROM clause must be distinct from each other.
   - Correlation names in a FROM clause must be distinct from the table or view names in the same clause.

   When the application contained ambiguities, such as

   ```
   SELECT A.COL1
   FROM A B, B A
   ```

   where COL1 appeared in both table A and table B, the system accepted the statement, employing its own set of rules to resolve the ambiguity. This example represents only one type of ambiguity that could occur.

   In V3R1, the ANS/ISO rules are fully enforced. Any violations generate SQLCODE -211 (SQLSTATE 52012).

2. New Reserved Words

   Prior to V3R1, the following were not reserved words in SQL and could therefore be used as ordinary identifiers:
   - CHAR
   - CHARACTER
   - DOUBLE
   - EXECUTE
   - FIELDPROC
   - GRAPHIC
   - LONG
   - PACKAGE.

   Similarly, the following were not reserved words for the DBS Utility:
   - REORGANIZE
   - SCHEMA.

   In V3R1, these are reserved words, so an existing application that uses any words in the SQL group above as an ordinary identifier will have to be changed before it is preprocessed, or SQLCODE -105 (SQLSTATE 37501) will be generated. Similarly, the words in the DBS Utility group above can no longer be used in DBS Utility commands as ordinary identifiers.

   You can address this incompatibility by changing these ordinary identifiers to use nonreserved words, or you can retain the original names by redefining them as delimited identifiers.

3. Significance of Trailing Blanks

Prior to V3R1, trailing blanks were treated as significant in both object names and VARCHAR and VARGRAPHIC column values.

In V3R1, such trailing blanks are not considered significant.

If your applications must continue to treat trailing blanks as significant, you may have to undertake some redesign. See “Detailed Notes on V3R1-V2R2 Incompatibilities” on page 399 for further discussion and examples.

4. Timestamp at the 24th Hour

   Prior to V3R1, a timestamp value in which the hour portion was 24 and the minute, second, or microsecond portion was not zero, was accepted as valid data for insertion or updating.

   In V3R1, an attempt to insert or update a column with such a value generates SQLCODE -181 (SQLSTATE 22007). When the hour portion is 24, the other time portions must now be zero.

   If you have any of these invalid values in your tables after migrating to V3R1, they will prevent you from doing a DBS Utility unload/reload operation or an INSERT using a subselect. You will have to first correct these values to conform to the rule mentioned above.

Application Programming

5. Invalid Pointers in SQLDA and RDIIN

   Prior to V3R1, the system checked for invalid pointers in the SQLDA and RDIIN structures. This checking was extensive, often resulting in poor performance.

   In V3R1, in the interest of better performance, this checking has been eliminated. It is up to the application programmer to follow the rules on setting pointers in the SQLDA, as outlined in the chapter “Using Dynamic Statements” in the V3R1 *Application Programming* manual. Pointers in the RDIIN must not be changed by the application. If your application does not satisfy these rules, the results will be unpredictable.

6. Continuation Characters in Fortran

   Prior to V3R1, the Fortran preprocessor ignored any continuation character located in front of an EXEC SQL on the same line, provided it was not part of an IF or ELSE statement — even though such coding was incorrect.

   In V3R1, the continuation character is acknowledged and the EXEC SQL is ignored.

7. Missing Comma in COBOL Continuation Lines

   Prior to V3R1, if you left out an intended comma from a list of parameters in an SQL statement embedded in a COBOL program (as illustrated below) and did not code a continuation character in the next line, the system would assume a continuation character and misinterpret the parameter list, giving potentially wrong results.

   ```
   SELECT *
   FROM T1
   WHERE COL1 IN ('AB'    <--- missing comma
                  'CD',   <--- no continuation character
                  'EF')
   ```

   In V3R1, this error is detected and reported at preprocessor time.

8. DROP PROGRAM Statement Containing Host Variables

   Prior to V3R1, the processing of a DROP PROGRAM statement that contained host variables required a specific section in the access module. (In this form of the statement, the name of the owner of the program or the name of the program or both are expressed as host variables.)

Note on New Terminology: As of V3R1, PACKAGE becomes the new reserved word for PROGRAM, the latter remaining as a synonym. Access modules are now referred to as packages. This new terminology is used below.

In V3R1, the host variable form of the DROP PACKAGE statement no longer requires a section in the package. All the information required to execute the statement is sent with the execution-time request. You will be affected if you have this form of the DROP PACKAGE coded in your application programs.

If the programs that use these packages are explicitly repreprocessed, they will have to be recompiled (or reassembled) and relinked in order to execute successfully. Otherwise, errors will result, since there will be fewer sections in the new package and this will cause a mismatch between section numbers in the RDIIN structure and the new package.

9. Data Type of String Constants

   Prior to V3R1, application programs that assumed that string constants have a data type of VARGRAPHIC because they are used in the context of GRAPHIC and VARGRAPHIC data, were accepted.

   In V3R1, such constants are considered to be VARCHAR, and if used in conjunction with GRAPHIC or VARGRAPHIC data will result in an error, such as SQLCODE -171 (SQLSTATE 53015) or SQLCODE -408 (SQLSTATE 53021).

   If the host language is COBOL, PL/I, or C, you should use explicitly coded graphic constants. See the section of the V3R1 *SQL Reference* manual that discusses graphic string constants.

10. New Options in CREATE PROGRAM Statement

    Prior to V3R1, when the following three options:

    ```
    ISOL({RR|CS|USER})
    DATE({ISO|USA|EUR|JIS|LOCAL})
    TIME({ISO|USA|EUR|JIS|LOCAL})
    ```

    were used in conjunction with an extended dynamic access module, the values for these options were determined when statements referencing the extended dynamic access module were executed. The values were set based on the corresponding preprocessing options of the program containing the extended dynamic statements.

    Note on New Terminology: As of V3R1, PACKAGE becomes the new reserved word for PROGRAM, the latter remaining as a synonym of the former. Access modules are now referred to as packages. This new terminology is used below.

    In V3R1, these options are added to the CREATE PACKAGE statement, so that they become preprocessing options. This means that their values are stored with the package itself, and are enforced when the sections of the package are executed. Consequently, your programs may now run at a different isolation level than they did in V2R2.

See “Detailed Notes on V3R1-V2R2 Incompatibilities” on page 399 for examples that illustrate how incompatibilities may arise as a result of this change.

11. Views Created from SELECT *

    Prior to V3R1, views created as SELECT * FROM T1 required no special attention when being migrated from release to release, even when columns had been added to table T1 ***after*** the creation of the view.

    In V3R1, a necessary change to the system now requires special attention in the above situation. The first time the system encounters such a view in an application, it attempts to rebuild the view, and fails with SQLCODE -835 (SQLSTATE 56049).

    To avoid this failure, drop and recreate the view before running the application on V3R1. Depending on how your application logic is coded, you may have to change that logic in order to handle the extra columns that were added to table T1. The best practice is to avoid the use of SELECT * for view creation, and specify the explicit columns that the application requires.

12. Semicolon Delimiter in SYSVIEW Table

    Prior to V3R1, when a view was created through the DBS Utility or by running a preprocessed program, the CREATE VIEW statement was inserted into column VIEWTEXT of catalog table SYSVIEWS with a semicolon delimiter.

    In V3R1, this delimiter is no longer included.

    If your application has a dependency on the existence of this delimiter in the SYSVIEWS table, you will need to change it accordingly.

13. Replacement of Error Message ARI0565E

    Prior to V3R1, error message ARI0565E was issued during preprocessing of Fortran programs whenever the input source contained no SQL statements that required creation of a package.

    In V3R1, this message is replaced by information message ARI0565I. In addition, related message, ARI0598I, dealing with the status of the package, is modified.

    This could impact any automated operating system facility that you may be using (for example, the VM Programmable Operator) to scan the message number and text.

14. Replacement of SQLCODE -150

    Prior to V3R1, an attempt to modify data through a view based on more than one table generated SQLCODE -150.

    In V3R1, this is replaced with SQLCODE +149 at preprocessor time, and SQLCODE -149 (SQLSTATE 53007) at run time.

15. New Positive SQLCODEs

    Prior to V3R1, a number of negative SQLCODEs and associated positive RDSCODEs were returned during preprocessing to indicate a warning situation.

    In V3R1, new positive SQLCODEs are returned instead, which correspond identically to the above negative SQLCODEs in code number and (in most cases) message text and explanation. If the error is not removed, the corresponding negative SQLCODEs will be issued at run time.

    See “Detailed Notes on V3R1-V2R2 Incompatibilities” on page 399 for a list of these new positive SQLCODEs.

System Environment

16. Uppercase and Mixed Case in Message Text

    Prior to V3R1, all message text was in uppercase for all the languages available in the product except German, which was available only in mixed case.

    Note: The uppercase applied to both English language offerings, AMENG and UCENG. It also applied to the English text embedded in the DBCS languages Japanese and Korean (for example, “FORCE”, “SQLEND”).

    In V3R1, the message text of three more languages is now changed to mixed case only. These languages are AMENG (the default language setting), Italian, and Spanish. If you are using any of these three languages and you have existing case-sensitive applications that scan for specific message text in uppercase only, you will have to modify them to detect lowercase as well. This could impact any automated operating system facility that you may be using for this purpose (for example, the VM Programmable Operator).

    An alternative approach (for English users only) to modifying your applications would be to specify UCENG instead of AMENG, through the SET LANGUAGE command.

17. Authorization for Changing System Catalog Tables

    Prior to V3R1, certain portions of the catalog could be updated, deleted, or inserted into, by any user with DBA authority.

    In V3R1, the number of columns in the catalog tables for which these changes are allowed is reduced.

    This change may affect the authorization of some of your applications. See Appendix E of the V3R1 *SQL Reference* manual for a list of the columns that can now be updated, deleted, or inserted.

18. Modification of Sample Tables and Applications

    Prior to V3R1, the sample tables shipped with the product consisted of five Manufacturing tables and four Organizational-project tables. The sample applications shipped with the product used the Manufacturing tables.

    In V3R1, the Manufacturing tables are not included, but can be installed optionally. The Organization-project tables are enhanced to provide more guidance on referential integrity and also consistency across the IBM relational database products. The enhancements include:
    - Two new tables
    - A new column in an existing table
    - Renaming of a table
    - Modification of a foreign key definition.

    The sample applications are now modified to use the enhanced Organization-project tables. They now issue a ROLLBACK instead of a COMMIT, so that they can be rerun without having to first restore the sample database.

    If you have any applications that use these tables, such as an online tutorial or a test package for new releases, you will need to upgrade them accordingly.

Detailed Notes on V3R1-V2R2 Incompatibilities

1. Significance of Trailing Blanks

   Prior to V3R1, delimited identifiers "TABLE1" and "TABLE1ƀ" would be considered two different tables, and VARCHAR values 'ABC' and 'ABCƀƀ' two different values, where 'ƀ' represents a blank character.

   In V3R1, in the case of the table names, the system would not accept the two tables because they now have identical names. In the case of the VARCHAR values, they are considered equal, except in a LIKE comparison. However, if specified at INSERT or UPDATE time, trailing blanks are included in the varying length string data stored in the database.

If your applications must continue to treat trailing blanks as significant, you may have to undertake some redesign. For example, prior to V3R1, if your table had a VARCHAR column, COLX, containing 'AAAƀƀƀ' and you wanted to select all values from COLX that were not equal to 'AAA', the following search condition would satisfy this requirement, because it would return value 'AAAƀƀƀ' along with any other values not equal to 'AAA':

```
WHERE COLX <> 'AAA'
```

In V3R1, value 'AAAƀƀƀ' does not get returned in the above example. This search condition must be redesigned in order to get the same results as in prior releases. One solution is:

```
WHERE COLX NOT LIKE 'AAA'
```

For more discussion on migration considerations for this item, see "Considerations for VARCHAR and VARGRAPHIC Compare" in the chapter which discusses migrating from V2R2, in the *System Administration* manual, V3R1 or later.

2. New Options in CREATE PROGRAM Statement

   The following examples illustrate the incompatibilities that may arise when you migrate to V3R1.

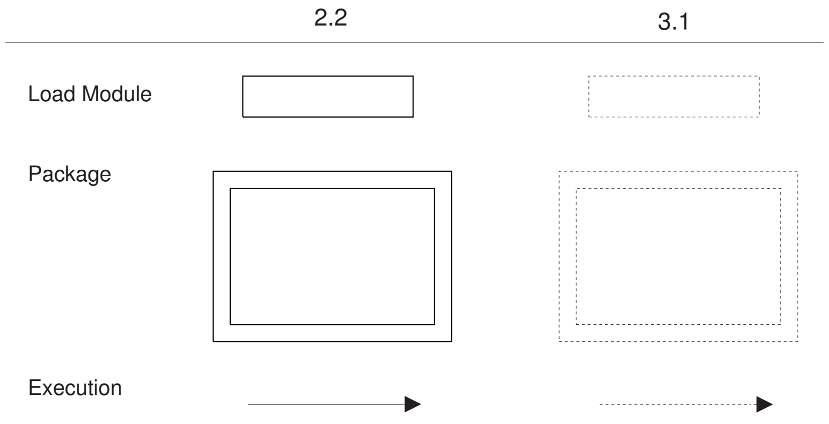


Figure 127. Legend

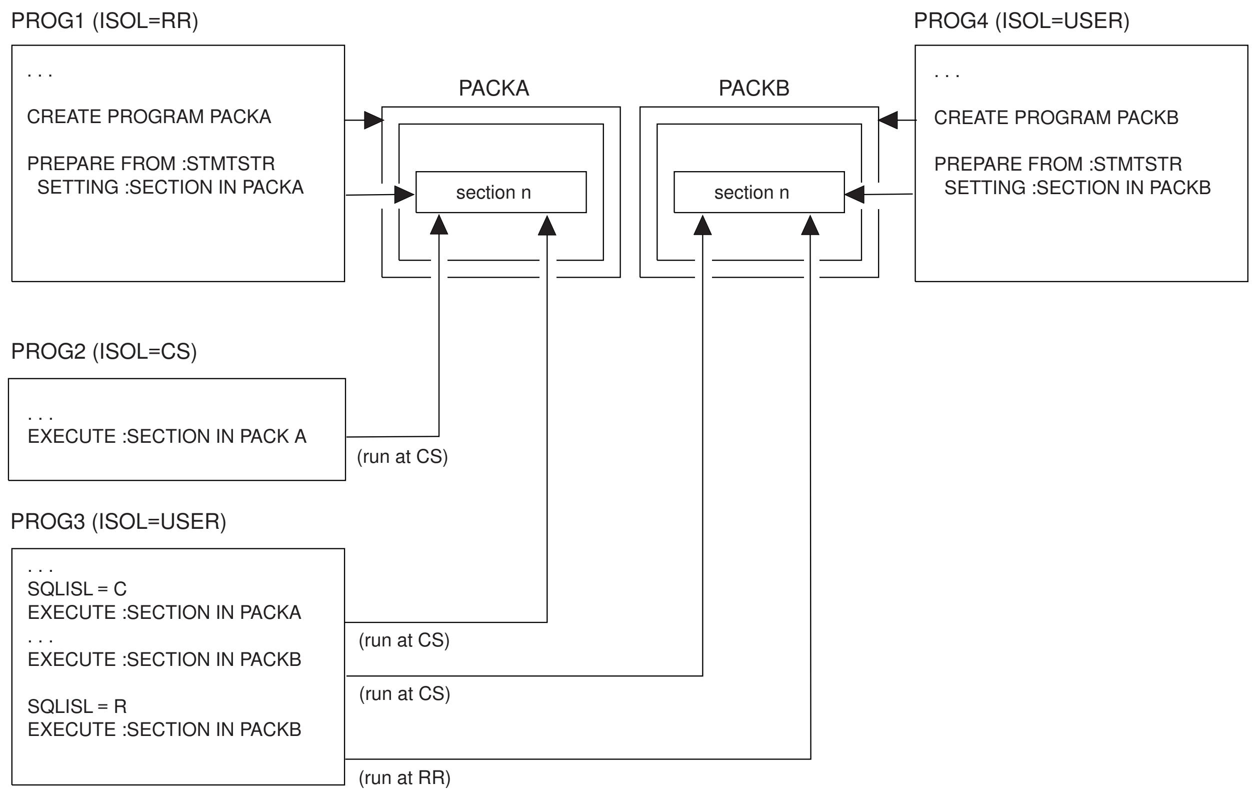


Figure 128. Version 2 Release 2

Figure 128 illustrates how isolation levels are determined for packages created using extended dynamic SQL in V2R2. For example, program PROG1 contains the CREATE PROGRAM statement for package PACKA, and prepares a section in the package. Program PROG2 subsequently executes the section in PACKA. Since program PROG2 was preprocessed with isolation level cursor stability (CS), the section executes using CS.

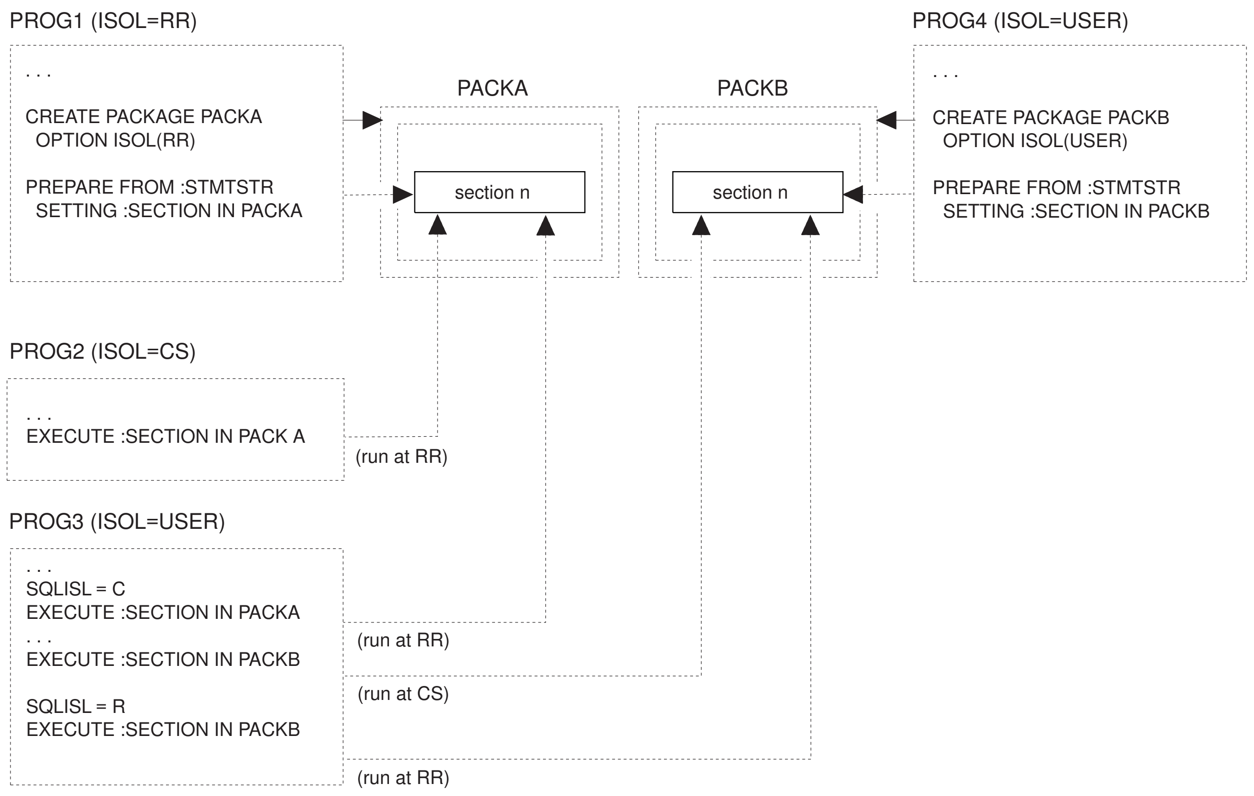


Figure 129. Version 3 Release 1

Figure 129 shows the same scenario in V3R1. In this case, the isolation level RR is specified when the PACKA package is created. When program PROG2 executes a section in PACKA, isolation level RR is used.

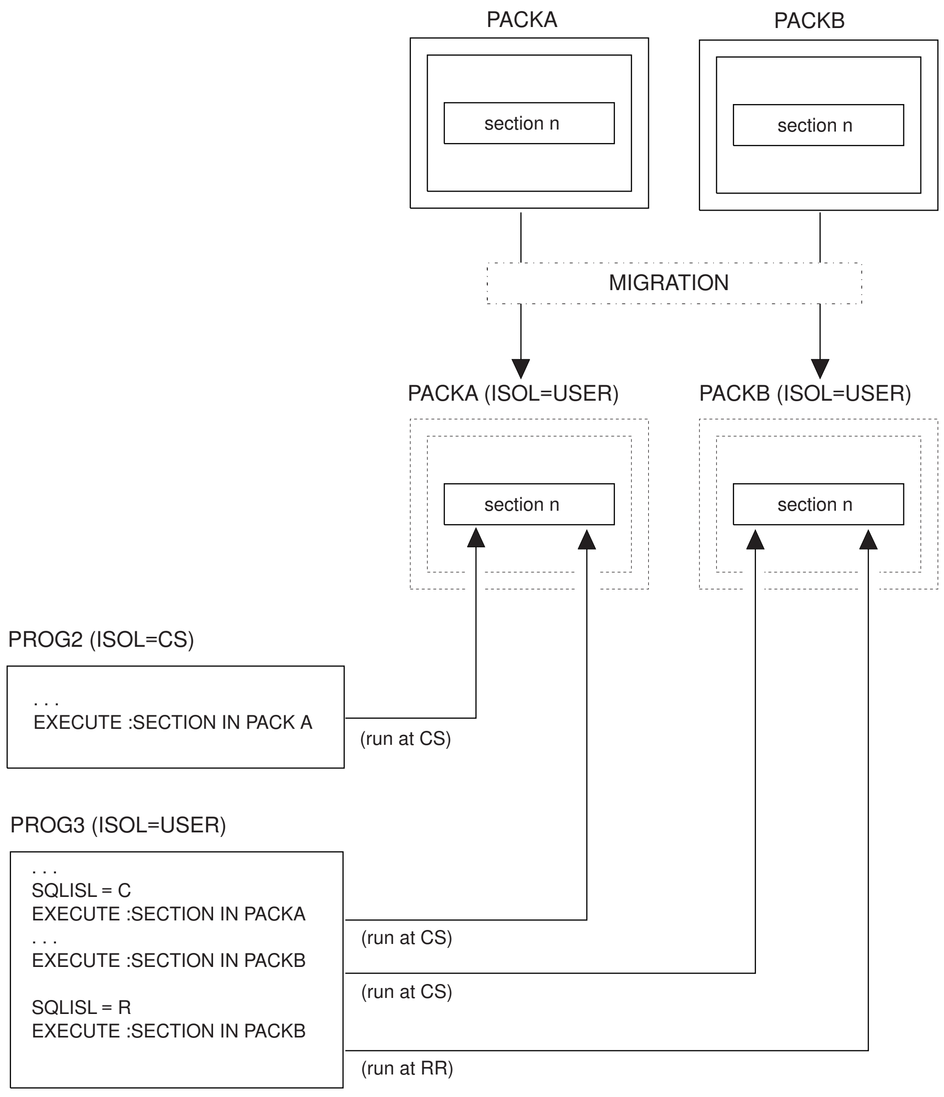


Figure 130. Migration

Figure 130 shows packages being migrated to V3R1. In this case, the isolation level bind option will be automatically set to USER. Applications will notice no change in isolation level handling from previous releases.

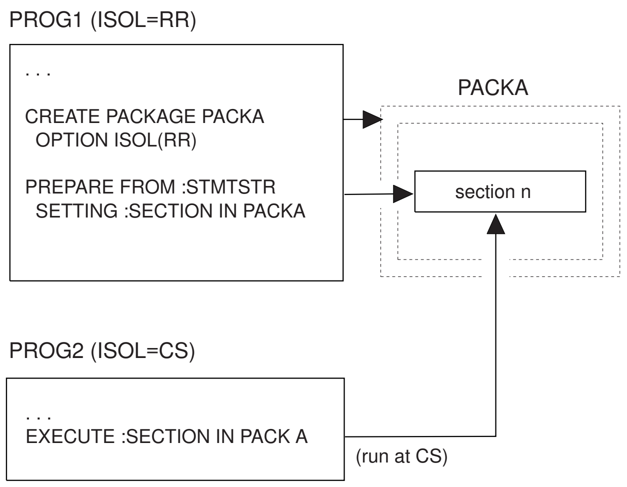


Figure 131. Dropping and Re-creating PACKA Without Repreprocessing PROG2

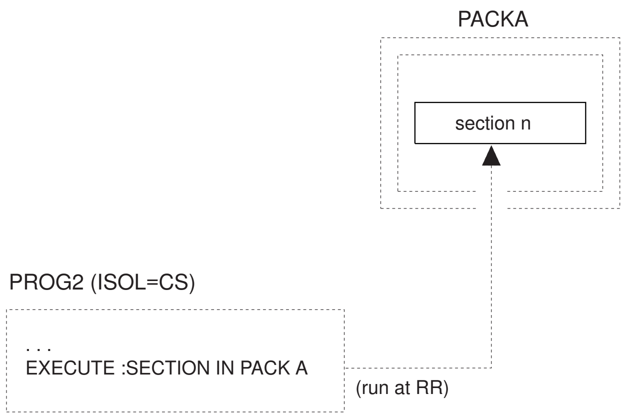


Figure 132. Re-preprocessing PROG2

Figure 131 and Figure 132 show that once an extended dynamic package has been dropped and recreated in V3R1 with an isolation level other than USER, the isolation level bind option will be enforced whenever the executing application has also been preprocessed, assembled, and re-linked under V3R1. If the PACKA package has been dropped and recreated in V3R1, with an isolation level of RR, then:

- If program PROG2 is still pre-V3R1, when the section in PACKA is executed, isolation level CS will be used.
- Otherwise, isolation level RR will be enforced whenever sections in PACKA are executed.

3. New Positive SQLCODEs

   These codes are shown in the table below.

SQLCODE	SQLSTATE	DESCRIPTION
+117	01525	The number of data values to be inserted does not equal the number of columns specified or implied.
+134		Improper use of long string.

SQLCODE	SQLSTATE	DESCRIPTION
+135		The input for a long string column in an INSERT statement or UPDATE statement must be from a host variable or be the keyword NULL.
+149		The view cannot be used to modify data because it is based on more than one table.
+151		A column of a view cannot be updated since it is derived from an expression.
+154		View limitations do not allow you to use the following operation: xxxxxx
+202	01533	Column xxxxxx was not found in any table referenced by the statement.
+204	01532	xxxxxx was not found in the system catalog.
+205	01533	Column xxxxxx was not found in table yyyyyy.
+206	01533	The xxxxxx on yyyyyy was not found.
+401		Incompatible data types found in an expression or compare operation.
+404		A character string specified in an INSERT or UPDATE statement is too large for the target column.
+405		The numeric value, xxxxxx, is not within the range of the data type.
+407		Either an UPDATE statement or an INSERT statement with a null value for a column defined as NOT NULL is not allowed, or a null host variable value is not allowed in a SELECT list.
+408		An UPDATE or INSERT of a data value is incompatible with the data type of the associated target column.
+414		The LIKE clause was used for a numeric or date/time column type. LIKE must only be used with character or graphic compatible columns.
+415		The corresponding columns, n, of the operand of a UNION or a UNION ALL do not have comparable column descriptions.
+416		You cannot specify a long string column in the SELECT clause of a UNION.
+419		The precision of the numerator and/or the scale of the denominator are too large for decimal division.
+421		A hexadecimal literal associated with a graphic compatible column in a predicate cannot have an odd length.
+551	01548	User xxxxxx does not have the yyyyyy privilege.
+552	01542	xxxxxx is not authorized to perform this statement.
+668		Table xxxxxx is inactive and you cannot access it.

V3R2 and V3R1 Incompatibilities

SQL and Data

1. Nonexposed Table Names

Prior to V3R2, nonexposed table names (those that have an associated correlation name in the FROM clause) could be referenced within the SQL statement containing such a name.

In V3R2, this is no longer the case. Any application code that makes such a reference will have to be changed to reference the associated correlation name instead. Otherwise, SQLCODE -201 (SQLSTATE 52003) will be generated.

For example, if both tables in the FROM clause

```
FROM TABLE1, TABLE2 A
```

have a column named DESCR, any reference in the query to this column for the second table would have to be written as A.DESCR, not TABLE2.DESCR, because TABLE2 is a nonexposed table name.

2. DISTINCT Column Functions in HAVING Clauses

   Prior to V3R2, a DISTINCT column function was allowed in conjunction with a dyadic operator in the predicate of a HAVING clause. (A dyadic operator is an operator having two operands.) For example, the following would be accepted as valid:

   ```
   SELECT JOB, AVG(SALARY), AVG(BONUS)
   FROM EMPLOYEE
   GROUP BY JOB
   HAVING AVG(DISTINCT BONUS) + 50 > 100
   ```

   In V3R2, as part of the product's compliance with SQL-89 in the introduction of unary minus in DISTINCT column functions, this code is no longer allowed. Using it will generate SQLCODE -112 (SQLSTATE 37507).

3. New Reserved Word, SOME

   Prior to V3R2, SOME was not a reserved word in SQL and could therefore be used as an ordinary identifier.

   In V3R2, SOME is a reserved word that is used in quantified predicates as a synonym for ANY, so any existing applications that use it as an ordinary identifier will have to be changed before they are preprocessed under V3R2. Id SOME is used as an ordinary identifier, SQLCODE -105 (SQLSTATE 37501) will be generated.

   You can address this incompatibility by changing this ordinary identifier to use a nonreserved word, or you can retain the original name by redefining it as a delimited identifier.

4. Comparing Character Data with Unquoted Numeric Data

   Prior to V3R2, applications that compared character data type columns to an unquoted numeric, represented invalid SQL code that was accepted. For example,

   ```
   WHERE C1 = 3
   ```

   where C1 was defined as CHAR(1).

   In V3R2, this is no longer accepted. Doing this comparison will generate SQLCODE -401 (SQLSTATE 53018).

5. CHAR Scalar Function with a Timestamp Argument

   Prior to V3R2, applications that used a second argument for the CHAR scalar function, when the first argument was a timestamp expression, represented invalid SQL code that was accepted. The second argument was ignored.

   In V3R2, this is no longer accepted. Using this argument will generate SQLCODE -171 (SQLSTATE 53015).

6. No Column Name in a Column Function Within a HAVING Clause

Prior to V3R2, applications that used a column function within a HAVING clause with no explicit column name in its argument, represented invalid SQL code that was accepted. For example:

```
HAVING MIN(1) > 30
```

In V3R2, this is no longer accepted. Using this function will generate SQLCODE -111 (SQLSTATE 56001).

7. Even-numbered Precision for Columns

   Prior to V3R2, columns that were specified with even-numbered precision were rounded up to the next odd-numbered precision, when creating or altering a table. For example, DECIMAL(6,2) became DECIMAL(7,2) at CREATE time.

   Similar rounding up is also performed for arithmetic expressions found inside statements. For example, the expression 99.9999/12*(12+3) will become 099.9999/12*(12+3) during processing.

   In V3R2, this rounding is no longer done. In the above example, any application code that relies on such rounding in order to store seven digits in the column will require a redefinition of the column to DECIMAL(7,2), if the table gets recreated in V3R2. Otherwise, one of the following error conditions (depending on where the mismatch between column and length of the variable occurs) will be generated: SQLCODE -302 (SQLSTATE 22003), SQLCODE -405 (SQLSTATE 53020), or SQLCODE -413 (SQLSTATE 22003).

   Arithmetic expression that relies on such rounding to obtain enough precision to accommodate the result of the calculation will need modification. In the above example, the 99.9999 in the expression 99.9999/12*(12+3) must be changed to 099.9999 in order to accommodate the result which is 124.99988. Otherwise, SQLCODE -802 (SQLSTATE 22003) will be generated.

   Date/time Durations: Date and time durations are specified as DECIMAL(8,0) and DECIMAL(6,0) respectively, but if stored in the database prior to V3R2, they became DECIMAL(9,0) and DECIMAL(7,0) columns. Because of this, V3R2 still accepts the odd-numbered precision for these durations, when they are used as input.

   Performance of Assembler Programs: Assembler does not support even-numbered precision. If such table columns are referenced in a predicate containing a comparative host variable in an Assembler program, the latter must be declared with a precision one higher than the column. This leads to inefficient processing. You should consider redefining such table columns to odd-numbered precision to avoid this reduction in performance.

8. Floating-point Ranges

   Prior to V3R2, there was a certain range of floating-point values that went beyond the allowable values for the database and if encountered, would generate SQLCODE -405 (SQLSTATE 53020).

   In V3R2, because of a necessary change in the checking algorithm for floating-point constants, the following two narrow ranges have been added to the original range and will now also trigger SQLCODE -405 when encountered:

   - Approximately +7.2370055773322608E+75 to +7.23700557733622E+75

- Approximately -7.2370055773322608E+75 to -7.23700557733622E+75.

9. Decimal Precision in Internal Sorts

   Prior to V3R2, an arithmetic operation involving decimal columns, such as COL1*COL2/100 or SUM(COL3), allowed a precision of up to 15 digits, unless the SQL query specified something less.

   In V3R2, with the enhancement of decimal precision, this allowable precision is now expanded to 31 digits. As a result, it is possible for a query that has been migrated from V3R1 to generate SQLCODE -101 (SQLSTATE 54001) with a value of 'ARIXECK' in the SQLERRP field of the SQLCA. This error indicates that the maximum allowable size (255) of an internal sort key has been exceeded. This can only occur if the query is fairly complex and requires an internal sort.

   Note: Queries that use internal sorts are typically those that use ORDER BY, UNION, or DISTINCT.

   If you experience this error, you can reduce the precision of the arithmetic operations in the select list of your query by applying the DECIMAL scalar function. This, in turn, may reduce the internal sort key to an acceptable length.

10. Quantified Predicates Involving Null Values

    Prior to V3R2, null values in quantified predicates (ALL, ANY) were not handled according to the FIPS standard.

    In V3R2, the FIPS standard applies. As a result, the truth value of these predicates is different from previous releases for some cases involving null values.

    See "Detailed Notes on V3R2-V3R1 Incompatibilities" on page 409 for a discussion on these cases and examples to illustrate the incompatibilities.

Application Programming

11. Negative Indicator Variables in Predicates

    Prior to V3R2, the use of negative indicator variables in predicates was limited to the basic equal-to (=) predicate in static, dynamic, and extended dynamic SQL.

    In V3R2, the use of negative indicator variables in predicates is extended in some areas and restricted in others. This use is now allowed in all predicates of static SQL and extended dynamic SQL when a descriptor is specified on the PREPARE statement. SQLCODE -309 (SQLSTATE 22512) is generated when a negative indicator variable is used in any predicate within dynamic SQL, or extended dynamic SQL when no descriptor is specified on the PREPARE statement.

12. Declaration of Indicator Variables

    Prior to V3R2, existing application programs that used indicator variables declared with a data type other than the equivalent of SMALLINT were accepted.

    In V3R2, these programs are no longer accepted. For FORTRAN programs, error message ARI0550E is generated at preprocessing time; for Assembler, C, COBOL, and PL/I programs, SQLCODE -326 (SQLSTATE spaces) is generated at preprocessor time.

13. Incorrect Data Inserted from Variable Length Host Variables

    Prior to V3R2, incorrect data could get inserted into the database from a variable length host variable that had a length value greater than the maximum that was defined at preprocessing time.

In V3R2, this is prevented. If it is attempted, SQLCODE -311 (SQLSTATE 22501) will be generated.

14. Incorrect String Representations of Date/time Values

    Prior to V3R2, incorrect string representations of date/time values generated errors at preprocessor time.

    In V3R2, warning messages are issued instead; then if the string representations are not corrected, they will result in errors at run time.

15. COBOL Host Variable Names

    Prior to V3R2, if a COBOL program contained a hyphen (-) in the declaration of a host variable name, this hyphen could be represented as an underscore (_) where the name was used within an SQL statement.

    In V3R2, the preprocessor no longer accepts this substitution within the program.

    If you have any such substitutions in your COBOL source code, they will have to be converted to hyphens before preprocessing under V3R2.

16. Validation of Host Variables

    Prior to Version 3, applications containing any of the SQL statements SELECT, SELECT INTO, UPDATE, INSERT, or DELETE, could be preprocessed from a user machine on one release of the product to a database machine on another release of the same version.

    In V3R2, there is a change to the validation of host variables for these statements. As a result, this preprocessing fails when the two releases involved are V3R1 and a later release of Version 3. To circumvent this problem, you must preprocess the application from a user machine at the same release level as the database machine on which you would like the package created before compiling, linking, and executing the application from the user machine.

Detailed Notes on V3R2-V3R1 Incompatibilities

1. Quantified Predicates Involving Null Values

   Those cases for which your applications will give different results than they did in earlier releases can be divided into three types, as described below. The accompanying examples are based on these two tables, where the question mark represents a null value:

   ```
   Table T1:   C1     C2        Table T2:   C3
               --     --                    --
                ?      1                     ?
                2      2                     2
   ```

   Recalling that a quantified predicate involves the structure

   ```
   <expression> <quantifier> <subquery>
   ```

   the three types can be described as follows:

   a. Prior to V3R2, when

      - The value of the expression is NULL, and
      - The subselect returns an empty set,

      the truth value of the quantified predicate was UNKNOWN.

      In V3R2, the truth value is TRUE if the quantifier is ALL, and FALSE if the quantifier is ANY.

      In the example below, the second row of T1 is returned by any release of the database manager, but the first row of T1 is only returned by V3R2.

```
SELECT * FROM T1
WHERE C1 > ALL (SELECT C3 FROM T2 WHERE C3 > 2)
```

b. Prior to V3R2, when
   - The quantifier is ALL, and
   - The subselect returns at least one NULL, and
   - There are no values in the result of the subselect for which the *implied* predicate (the predicate applied to just one value in the result) is FALSE;

   or when
   - The quantifier is ANY, and
   - The subselect returns at least one NULL, and
   - There are no values in the result of the subselect for which the *implied* predicate (the predicate applied to just one value in the result) is TRUE,

   the truth value of the quantified predicate was FALSE, except when the expression was NULL.

   In V3R2, the truth value is UNKNOWN.

   Note: This change will only affect the results of queries in which a NOT has been applied to the quantified predicate in the situations described above. When a NOT is applied, the truth value is TRUE for prior releases, but is UNKNOWN for V3R2.

   In the example below, both rows of T1 are returned by previous releases, but only the first row of T1 is returned by V3R2:

   ```
   SELECT * FROM T1
   WHERE NOT C2 = ALL (SELECT C3 FROM T2)
   ```

   See the following references for performance implications of queries similar to those shown in the above examples:
   - Chapter 2 of the V3R2 *Database Administration* manual for a discussion on nulls in quantified predicates where null columns are allowed, under "Creating Tables".
   - Chapter 5 of the V3R2 *Diagnosis Guide and Reference* manual for a discussion on inefficient search where nullable expressions are involved, under "Analysis of Performance Problems".

c. Prior to V3R2, when:
   - The expression contains an arithmetic expression, scalar function or column function
   - The quantifier is ALL
   - The subselect returns at least one NULL, and
   - There are no values in the result of the subselect for which the *implied* predicate (the predicate applied to just one value in the result) is FALSE,

   the truth value of the quantified predicate was TRUE, except when the expression was NULL.

   In V3R2, the truth value is UNKNOWN.

   In the example below, the second row of T1 is not returned by any release of the database manager. However, the first row of T1 is returned by previous releases, but not by V3R2.

   ```
   SELECT * FROM T1
   WHERE C2 + 1 = ALL (SELECT C3 FROM T2)
   ```

Comparison of types (b) and (c): Type (c) is really a subset of the more general case outlined in type (b), by virtue of its extra condition about the expression. However, type (c) is included separately here, because it represents an ***exception*** to the more general case. The exception lies in the fact that these two types generated different results for the truth value prior to V3R2. In V3R2, however, this exception disappears, because their results are now the same (truth value = UNKNOWN).

V3R4 and V3R2 Incompatibilities (VSE Only)

SQL and Data

1. New Reserved Word, CONCAT

   Prior to V3R4, CONCAT was not a reserved word in SQL and could therefore be used as an ordinary identifier.

   In V3R4, CONCAT is a reserved word, and can be used as an alternative to the concatenation operator (||). Any existing applications that use it as an ordinary identifier will have to be changed before they are preprocessed under V3R4; otherwise SQLCODE -105 (SQLSTATE 37501) will be generated.

   You can address this incompatibility by changing this ordinary identifier to use a nonreserved word, or you can retain the original name by redefining it as a delimited identifier.
2. REVOKE UPDATE

   Prior to V3R4, the REVOKE statement for the UPDATE privilege ignored any column names that might be present as parameters of the UPDATE option — even though such coding was invalid. (This statement is only done on a table basis, never a column basis.)

   In V3R4, such parameters are not allowed. If they are used, SQLCODE -105 (SQLSTATE 37501) will be generated.
3. Numeric Data in Character Strings

   Prior to V3R4, columns with a data type of CHAR or VARCHAR accepted numeric data, including FLOAT, on insert or update. For example, the following statements did not create an error:

   ```
   CREATE TABLE T1 (COL CHAR(8))
   CREATE TABLE T2 (COL VARCHAR(8))

   INSERT INTO T1 (123)
   INSERT INTO T2 (123)
   INSERT INTO T1 (1E1)
   INSERT INTO T2 (1E1)

   UPDATE T1 SET COL = 123
   UPDATE T2 SET COL = 123
   UPDATE T1 SET COL = 1E1
   UPDATE T2 SET COL = 1E1
   ```

   In V3R4, these inserts and updates now generate SQLCODE -408 (SQLSTATE 53021).

   If you want to use the value 123, you must now use it as a character literal ('123'). Float literals are no longer allowed for character columns.
4. Invalid String Representation of Datetime

   Prior to V3R4, when a predicate was being evaluated that contained an operand that was one of the special registers CURRENT DATE, CURRENT

TIME, or CURRENT TIMESTAMP, and one of the other operands was a character column of the correct length but containing a value that was not a valid string representation of a datetime, the application ran successfully. Any row containing such an invalid value was returned if it met the search condition. For example, all invalid date values in column, ORDERDATE, were returned for the following condition:

```
WHERE CURRENT DATE <> ORDERDATE
```

In V3R4, SQLCODE -180 (SQLSTATE 22007) is generated under the above condition.

5. Internally Generated Table Names

   Prior to V3R4, the system internally built a composite table name that included the name of the relational database, based on a certain maximum length.

   In V3R4, this length is slightly increased, and the internal process is the same, whether DRDA server support is involved or not. As a result, there is a very small probability that some of your SQL statements could exceed an internal limitation of the system and generate an SQLCODE -101 (SQLSTATE 54001).

   The more table names you have in a statement, the greater the probability of this occurring. If you experience this error, one possible solution would be to break the statement down into two separate statements.

6. Enhanced EXPLAIN Tables

   Prior to V3R4, the tables used by the EXPLAIN statement had some major differences from the corresponding tables in the DB2* product.

   In V3R4, these differences are minimized to enhance the EXPLAIN functions and make them more compatible with those in the DB2 product. As a result, there are significant changes to the design of these tables and the EXPLAIN statement no longer works on the old tables. These changes include new columns dispersed among old ones, the loss of one column, a column data type change, and a column length change.

   See the *DB2 Server for VSE & VM SQL Reference* manual for the new design of these tables.

   If you have used the EXPLAIN tables in prior releases, you will have to recreate the revised tables before using the EXPLAIN statement in V3R4. To assist you in this task, a DBSU job file containing the necessary create statements is now included as an A-type member (called ARIXEXP) with the product.

   Similarly, if you have applications which depend upon the design of the old EXPLAIN tables, you will need to modify these applications to reflect the new design.

Application Programming

7. Setting of SQLN Field

   Prior to V3R4, if field SQLD in the SQLDA area held a greater value than the SQLN field after a DESCRIBE, the system set SQLN to zero.

   In V3R4, the value of SQLN is not changed.

   If your application tests SQLN for zero to verify successful completion of the DESCRIBE, the logic will have to be revised to test for SQLD > SQLN.

8. C NUL-Terminated Strings - Variable Length

   Prior to V3R4, a C input string with a length greater than 1 was treated as a fixed length character host variable. It was not mandatory to have a NUL present in it except when the input host variable length was 255, in which case SQLCODE -426 (SQLSTATE 22523) was generated.

In V3R4, a C input string is no longer treated as fixed length. A NUL must be present on all C NUL-terminated input strings except those with a length of 1; otherwise SQLCODE -302 (SQLSTATE 22001) is generated. SQLCODE -426 (SQLSTATE 22523) is no longer generated.

9. C NUL-Terminated Strings - NUL Byte

   Prior to V3R4, the NUL byte in a C NUL-terminated string was treated as a blank.

   In V3R4, it is treated as a string terminator.

10. C NUL-Terminated Strings - Trailing Blanks

    Prior to V3R4, any trailing blanks in a C NUL-terminated string were removed when using the string to update or insert a VARCHAR column or to compare to a VARCHAR column.

    In V3R4, these blanks will no longer be removed.

11. C NUL-Terminated Strings - Length

    Prior to V3R4, the SQL/DS scalar function, LENGTH, with a C NUL-terminated string as its argument, returned the defined length.

    In V3R4, this function now returns the length according to the position of the NUL terminator. (This length excludes the terminator itself.)

12. SQL Statement String

    Prior to V3R4, an SQL statement string could end with a statement terminator, when used in conjunction with EXECUTE IMMEDIATE, PREPARE, or Extended PREPARE. An example of such a statement is

    ```
    DROP TABLE T1;
    ```

    which has a trailing semicolon. This was allowed in application programs, even though such coding was invalid. It was also allowed in ISQL and QMF*, since those facilities also use the above three statements to process interactively issued statements.

    In V3R4, this statement terminator is not allowed. If it is used, SQLCODE -104 (SQLSTATE 37501) will be generated.

    If you have been using such a terminator for the CREATE VIEW statement, your use of catalog table SYSVIEWS could be affected, as described in the *DB2 Server for VSE & VM SQL Reference* manual.

13. SQL/DS Preprocessing of Extended Dynamic Statements

    Prior to V3R4, a cursor-variable with a defined length greater than 18 was accepted by the preprocessor, even though such variables should only be defined with a length of 18.

    In V3R4, the preprocessor traps this condition and generates SQLCODE -324 (SQLSTATE spaces). You will have to change any applications that use these invalid cursor-variable lengths in your extended dynamic statements.

14. Reason Codes for Incorrect Host Variable Declarations

    Prior to V3R4, a large number of SQLERRD1 codes were associated with SQLCODE -314 (SQLSTATE spaces) at preprocessor time for invalid host variables.

    In V3R4, with the introduction of host structures and the associated parsing of declaration statements by the preprocessor, the values of some of these SQLERRD1 codes have changed.

    If your application has dependencies on specific SQLERRD1 values, you should look for these changes in the *DB2 Server for VM Messages and Codes* or *DB2 Server for VSE Messages and Codes* manual and modify your application accordingly.

15. Structured Declarations in COBOL and C

    Prior to V3R4, there were a number of error situations for structure declarations in the SQL DECLARE SECTION that were not checked by the COBOL and C preprocessors.

    In V3R4, these situations are subjected to validation checks, resulting in the following potential errors, which must be corrected before compilation:

SQLCODE	SQLSTATE	Condition
-107	54003	Host variable name too long
-307	spaces	Duplicate host variable names
-314	spaces	Syntax and semantic errors in a host variable

16. Data Type of Hexadecimal Constants

    Prior to V3R4, application programs that assumed that hexadecimal constants have a data type of VARGRAPHIC, because they are used in the context of GRAPHIC and VARGRAPHIC data, were accepted.

    In V3R4, such constants are considered to be VARCHAR. If used in conjunction with GRAPHIC or VARGRAPHIC data, they will cause a number of specific SQLCODEs and corresponding SQLSTATEs, dependent on individual cases.

    This also means that SQLCODE -421 (SQLSTATE 53055), dealing with hexadecimal literals of odd length, is no longer generated.

17. Non-updatable View

    Prior to V3R4, a user with DBA authority who tried to update a view that was not updatable got an appropriate error, such as SQLCODE -154 (SQLSTATE 56009). A user without DBA authority, however, got an authorization error, SQLCODE -551 (SQLSTATE 59001).

    In V3R4, the latter user receives the same error message as the DBA user, instead of the authorization message.

18. SYSTEM Table Missing from the System Catalog

    Prior to V3R4, if you tried to INSERT, DELETE, or UPDATE a table or view created by 'SYSTEM', but which was not in the system catalog, SQLCODE -823 (SQLSTATE 53032) was generated, indicating that you lacked proper authorization.

    In V3R4, SQLCODE -204 (SQLCODE 52004) is generated instead, indicating that the object could not be found in the system catalog.

19. Folding of Lowercase in PREP and DBSU

    Prior to V3R4, folding of lowercase into uppercase in PREP and the DBS Utility was done by adding X’40’ to the hexadecimal representation of the lowercase character. Sometimes this resulted in characters being folded incorrectly (for example, in the Katakana character set).

    In V3R4, this is done using the 370 built-in Assembler instruction TRANSLATE and the user-specified character translation table, in order to be consistent with how the application server handles this operation. One exception to this is when the DBS Utility processes SCHEMA input files. Folding is no longer done on these files; this makes it consistent with the DBS Utility control file, which only allows uppercase input.

    If your applications have built-in dependencies on the previous folding scheme, you could get different results. For example, a Katakana user may have a character in his or her coding scheme that has a hexadecimal value that appears to the SQL/DS system as one of the 26 lowercase English letters.

Instead of being folded to uppercase English, the Katakana character will now be folded according to the Katakana character translation table.

If you have lowercase in your DBS Utility SCHEMA input file, you will have to change it to uppercase.

20. Loading Audit Trace

    Prior to V3R4, the *Database Administration* manual contained sample table definition and DATALOAD parameters for creating a security audit table and loading trace records into it.

    In V3R4, the position of the columns within the table are changed and a new column, EXTLUWID, added. If you have been loading audit trace data using this table definition and a DATALOAD job, you will need to change the DATALOAD job, as documented in the V3R4 *Database Administration* manual. If you also want to make use of the new EXTLUWID column, you will need to recreate the table as well.

21. Use of Host Variables in CONNECT Statement

    Prior to V3R4, if you used a host variable for the userid or password in a CONNECT statement and the data type of that variable did not satisfy one of the conditions listed below, an error was generated at run time:
    - C programs: C-NUL string of length 9
    - Assembler, COBOL, or PL/I programs: fixed length character string of length 8.

    In V3R4, these conditions are checked by the preprocessor. If they fail the check, SQLCODE -324 (SQLSTATE spaces) is generated.

22. Data Types of Parameter Markers in Predicates

    Prior to V3R4, the resolution of data types for a parameter marker was dependent on the highest order of the data types of all the operands to the left of the parameter marker. Highest order, in the case of numeric operands, implies FLOAT > DECIMAL > INTEGER > SMALLINT.

    In V3R4, this resolution process is changed to become more consistent with the DB2 product. If there is an operand expressed as a column name in a BETWEEN predicate, the data type of any parameter marker is resolved as that of the leftmost such operand. Otherwise, the data type of the parameter marker is resolved as that of the leftmost operand that is not a parameter marker — whether in a BETWEEN predicate or an IN predicate.

    This could cause a different result from previous releases for predicates that can have more than two operands (namely BETWEEN and IN), but only if your application assigns parameter marker values that are inappropriate for your data.

    See “Detailed Notes on V3R4-V3R2 Incompatibilities” on page 418 for some examples and further discussion.

23. Bad Input Records in DATALOAD

    Prior to V3R4, a bad input record would terminate DATALOAD command processing on multiple tables when the DBS Utility was running in multiple user mode — whether or not it was preprocessed with the NOBLOCK option. An insert error would be indicated with one of the following codes, followed by message ARI0862E:

SQLCODE	**SQLSTATE**
-405	53020
-424	22502
-530	23503
-802	22003, 22012, or 22502
-803	23505

In V3R4, such command processing is no longer terminated, if the DBS Utility is preprocessed with the NOBLOCK option. The error indications are still generated, but the processing skips over the bad record and continues.

If you have a dependency in your application on this termination approach prior to V3R4, you may want to address this change in the case of the NOBLOCK option.

24. Index Dependency of a Package

    Prior to V3R4, when a SELECT DISTINCT was applied to a single column that had a unique index, the system assumed uniqueness within the column, rather than applying a sort. However, this kind of index dependency was not recorded in the package.

    In V3R4, this technique now records the index dependency in the package (for system integrity), even though the index is not actually used to access the table. In addition, the technique is extended to column functions that use DISTINCT — for example, SELECT COUNT(DISTINCT(COL4)), where COL4 has a unique index.

    If the index is dropped, the package will now be marked as invalid, causing a dynamic reprep. After the reprep, the application will take longer to execute, because a sort will be needed to process DISTINCT correctly.

25. SQLSTATE Changes

    Prior to V3R4, certain SQLCODEs had associated SQLSTATEs that did not conform to the SAA standards.

    In V3R4, these SQLSTATEs are replaced with ones that do conform. See "Detailed Notes on V3R4-V3R2 Incompatibilities" on page 418 for a list of these codes, along with their old and new SQLSTATEs.

System Environment

26. The Use of DBCS Characters with the CHARNAME Setting

    Prior to V3R4, you could use graphic or mixed constants, the VARGRAPHIC scalar function, or you could define columns as GRAPHIC or FOR MIXED DATA, independent of the CHARNAME setting on the application server. Furthermore, you could use graphic or mixed constants, independent of the CHARNAME setting on the application requester.

    In V3R4, the above usages result in error conditions such as SQLCODE -640 (SQLSTATE 56031) and SQLCODE -332 (SQLSTATE 57017), if the corresponding CHARNAME does not define a character set with mixed CCSID (that is, if CCSIDMIXED = 0).

27. Setting of CHARNAME

    Prior to V3R4, if no CHARNAME is specified, SQLSTART defaulted to CHARNAME = ENGLISH.

    In V3R4, it defaults to the CHARNAME used on the previous invocation. If the CHARNAME setting does not define a character set with mixed CCSID (that is, if CCSIDMIXED = 0), then the default character subtype (CHARSUB) will be forced to a value of SBCS.

    See the V3R4 *System Administration* manual for the initial default CHARNAME value after installation or migration.

28. Addressing Mode 31-Bit

    Prior to V3R4, user exits and field procedures , executed in a VSE environment, only ran in 24-bit addressing mode.

    In V3R4, with VSE/ESA* 1.3 or later releases, they can be executed in 31-bit addressing mode. If the SQL/DS system is running in 31-bit addressing mode (that is, ESA or VMESA supervisor mode) on the application server, then user exits (except accounting) will be executed in 31-bit addressing mode.

If you have user exits (except accounting) that fit into this category, you must do one of the following to avoid any potential problems:

- Ensure that they can accommodate 31-bit addressing mode
- Operate the application server in 370 or VM supervisor mode.

For more information on user exits, see the *DB2 Server for VSE System Administration* manual.

29. Section Size in a Package

    Prior to V3R4, during the preprocessing of a program, the system allocated a section size for each statement in the package.

    In V3R4, due to other design changes, it is necessary to increase the size of these sections for SELECT statements. As a result, when an existing package is subjected to a dynamic repreparation, it may cause the dbspace to become full, generating SQLCODE -946 (SQLSTATE 57025).

    If this occurs in your installation, you will have to explicitly prepare the program with the SQLPREP EXEC, making sure that you have a dbspace that can accommodate the revised package.

    Also, the larger sections increase the amount of virtual storage required to run the package. For example, if you have many dynamic SELECT statements in a logical unit of work, they will use up more storage than in the previous release.

30. Three-Part Object Names

    Prior to V3R4, an object that was created on a database named (for example) DBX could be successfully referenced later by an application, even though the name for that database had been changed (to, say, DBY). All you had to do was use the revised name, DBY, when you established the database for the application.

    In V3R4, the system maintains the name of the database that was used at the time of the object's creation (DBX in this example), as the first part of the object name, thereby making it a three-part name. If you now establish the database for the application under a different name (for example, DBY), the system uses that name as the new qualifier when you try to reference the object. This results in a mismatch of object names, and causes SQLCODE -114 (SQLSTATE 56061) to be generated.

    This problem can be avoided by simply not changing the names of your databases.

31. Special Characters for CONCAT Operation and Not Equal Condition

    Prior to V3R4, the class of the hexadecimal values in the table below was 0.

CHARNAME	**Hexadecimal Values**
ENGLISH	X'5A', X'B0'
FRENCH	X'BA', X'BB'
GERMAN	X'BA', X'BB'
ITALIAN	X'BA', X'BB'
KATAKANA	X'5A', X'B0'
SPANISH	X'BA', X'BB'

In V3R4, the class of these hexadecimal characters is changed to 6. This is reflected in the CHARCLASS column values of the SYSTEM.SYSCHARSETS catalog table. This change provides additional special characters that can be used to depict the CONCAT operation and the not equal condition in SQL syntax. This, in turn, provides greater flexibility in the use of these two SQL facilities between application requesters and servers that are assigned different CHARNAMES.

This could affect your applications, if they are dependent on previous reclassifications of any of the above characters from class 0 to class 3, for use in ordinary identifiers. For example, if you had reclassified the explanation mark (!) so that DANGER! could be used as an ordinary identifier, this will no longer work because the explanation mark is one of the characters that is now assigned to class 6.

See the *DB2 Server for VSE System Administration* manual for details on these classifications.

32. Invocation of TRACE for Storage

    Prior to V3R4, if you specified level 2 trace for the STAT or PA component of the TRACDBSS or TRACRDS parameter, respectively, when starting the SQL/DS system, you received the Working Storage Manager tracing.

    In V3R4, you can use the same specifications, but the Working Storage manager tracing is no longer part of the output.

    In order to get this part, you must now use the TRACSTG parameter, or select the STG component when using the TRACE operator command. The format from this trace is different.

33. Change to Headers in Multiline Operator Console Messages

    Prior to V3R4, for ease of reading, only the first line of a multiline message contained the message header identification, as illustrated below:

    ```
    ARI0418A SQL/DS is not ready. Retry the enable
             transaction CIRB after SQL/DS starts.
    ```

    However, operator console messages which were multiline could not be handled by the VSE Programmed Operator tool, because the system sent such messages one line at a line. The tool could not identify the extra lines.

    In V3R4, these operating console messages are sent as one multiline record, so that the VSE Programmed Operator tool can handle them. (For the console operator, there is no change to the appearance of these messages.)

    If you have your own application equivalent to the above tool, it could be affected by this change.

Detailed Notes on V3R4-V3R2 Incompatibilities

1. Data Types of Parameter Markers in Predicates

   In this first example, prior releases would resolve the data type of the parameter marker as DEC(4,2), whereas V3R4 would resolve it as INTEGER (assuming INTEGERCOL is the name of a column with a data type of INTEGER).

   ```
   23.55 BETWEEN ? AND INTEGERCOL
   ```

   The next two examples illustrate how these data type differences can produce quite different end results when the SQL statement is executed. In this next example, the predicate would generate SQLCODE -302 (SQLSTATE 22003) in prior releases, when the leftmost parameter marker is assigned a value of 345 and the rightmost parameter marker is assigned a value of 206.7. This error will not occur in V3R4.

   ```
   EDLEVEL IN (16, ?, 17.3, ?)
   ```

This is because the prior releases assign a data type of DEC(3,1) to the rightmost parameter marker, to which the value 206.7 cannot be assigned. V3R4 assigns a data type of SMALLINT to the rightmost parameter marker (based on the column EDLEVEL) and then truncates 206.7 to accommodate this data type.

In the next example, the predicate would generate SQLCODE -302 (SQLSTATE 22001) in V3R4, but not in prior releases, when the parameter marker is assigned a value of 'GHIJKL'.

```
DEPTNO IN ('ABCDEF', ?, 'ABC')
```

This is because V3R4 assigns a data type of CHAR(3) to the parameter marker (based on column DEPTNO), to which the value 'GHIJKL' cannot be assigned. Prior releases assign a data type of CHAR(6) to the parameter marker.

2. SQLSTATE Changes

   These changes are shown in the following table.

SQLCODE	Old SQLSTATE	New SQLSTATE	DESCRIPTION
-131	53004	22019	Either the LIKE predicate has an invalid escape character, or the string pattern contains an invalid occurrence of the escape character.
-551	59001	42501	User wwwwww does not have the xxxxxx privilege to perform yyyyyy on zzzzzz.
-552	59002	42502	xxxxxx is not authorized to yyyyyy.
-554	59002	42502	You cannot grant a privilege to yourself.
-555	59002	42502	You cannot revoke an authority or a privilege from yourself.
-556	59002	42502	An attempt to revoke a privilege from xxxxxx was denied. Either xxxxxx does not have this privilege, or yyyyyy does not have this authority to revoke this privilege.
-556	59004	42504	An attempt to revoke a privilege from xxxxxx was denied. Either xxxxxx does not have this privilege, or yyyyyy does not have this authority to revoke this privilege.
-558	59004	42504	You cannot revoke an authority from xxxxxx because xxxxxx has DBA authority.
-560	59005	42505	A CONNECT statement contains an incorrect password for xxxxxx.
-561	59005	42505	User xxxxxx does not have CONNECT authority.
-566	59001	42501	User ID xxxxxx does not have authorization to modify package yyyyyy.
-606	59002	42502	The COMMENT ON or LABEL on statement failed because the specified table or column is not owned by xxxxxx.
-610	59002	42502	The statement failed because a user without DBA authority attempted to create a table in a DBSPACE owner by another user or by the system.
-708	59002	42502	You cannot ALTER, LOCK, or DROP a PUBLIC DBSPACE because you do not have DBA authority.
-713	37515	53015	Incorrect isolation level value xxxxxx specified. Only values C or R may be used.
-801	22004	22003	Exception error xxxxxx occurred during yyyyyy operation on zzzzzz data.
-802	22004	22003	Exception error xxxxxx occurred during yyyyyy operation on zzzzzz data, position nnnnnn. psw1 psw2.

SQLCODE	Old SQLSTATE	New SQLSTATE	DESCRIPTION
-815	59005	42502	CONNECT denied by accounting user exit routine.
-30053	59006	42506	Owner xxxxxx authorization failed.

V3R5 and V3R4 Incompatibilities

1. SQL/DS Database Archive Incompatibilities

   Archives that were created on prior releases of SQL/DS cannot be restored by the SQL/DS V3R5 database manager. If this is attempted, the database manager will issue message ARI2038E and terminate. See the *DB2 Server for VM Messages and Codes* or *DB2 Server for VSE Messages and Codes* manual for more details on this message.
2. SQL/DS VSAM Shareoptions Changes under VSE

   In prior releases of SQL/DS (VSE), the VSAM SQL/DS directory, data and log data sets were defined with SHAREOPTIONS(1). In SQL/DS V3R5, these VSAM files must now be defined with SHAREOPTIONS(2).
3. SQLSTATE Values Changes

   Many SQLSTATE values have changed in SQL/DS V3R5. The new SQLSTATE values and their former values can be found in the *DB2 Server for VM Messages and Codes* or *DB2 Server for VSE Messages and Codes* manuals. Changing SQLSTATEs is an incompatible change since many SQLSTATE values that are returned from diagnostic situations will be different from previous releases of SQL/DS. Application programmers should review any programs that use SQLSTATE in the SQLCA each time an SQL statement is executed.
4. Messages and Codes Changes

   Some SQL/DS messages and codes have changed, and some new ones have been added in SQL/DS V3R5. See the *DB2 Server for VM Messages and Codes* and *DB2 Server for VSE Messages and Codes* manuals for details.
5. Display CICS Information on SHOW CONNECT

   If the package that the connected user is running was created in SQL/DS Version 2 Release 2 or earlier, the CICS information will not be displayed by the SHOW CONNECT command because the RDIIN for V2R2 or earlier does not contain the RDIIN extension area. The package must be reprepped with SQL/DS V3R5 and recompiled to make the CICS information available.

V5R1 and V3R5 Incompatibilities

1. Messages and Codes Changes

   Many messages and codes have changed, and some new ones have been added in DB2 Server for VSE & VM Version 5 Release 1. See the *DB2 Server for VM Messages and Codes* and *DB2 Server for VSE Messages and Codes* manuals.
2. DB2 Database Archive Incompatibilities

   Archives that were created on prior releases cannot be restored by the DB2 Server for VSE & VM Version 5 Release 1 database manager. If this is attempted the database manager will issue message ARI2038E and terminate. See the *DB2 Server for VSE Messages and Codes* manual.
3. DBSU

   If you use R350 DBSU to unload and reload a table in a R510 database, the value of the DATACAPTURE column will be lost.
4. Date/Time Exits and Field Procedures

VM Users with Date/Time or Field Procedure Exits that are dependant on running in a 370 Mode virtual machine must convert to execute in a ESA mode virtual machine. Note that exits requiring AMODE=24 are not affected, as we still support running the Server code in AMODE=24. The above also applies to Single User Mode application programs. The above also applies to Vendor programs that run on the Server, such as database monitoring or tape mount handling programs.

V6R1 and V5R1 Incompatibilities

1. Exploiting RDS above the 16 Megabyte Line

   With Version 7 Release 5, the RDS component is linkedited with the ″RMODE ANY″ option. This allows RDS to be loaded and executed above the 16MB line. This will free up valuable storage below the 16 MB line. As the RDS code will be loaded above the 16MB line before other storage is allocated, extremely storage constrained systems may need to increase their partition size to maximize their below the 16MB line free storage.
2. DBNAME Directory format change

   The format of the DBNAME directory source member, ARISDIRD, has been changed to support DRDA Online Requester support. See "Setting Up the DBNAME Directory" on page 23. See the *DB2 Server for VSE System Administration* manual.

V7R1 and V6R1 Incompatibilities

1. DBNAME Directory format change

   ARISDIRD has been restructured to improve readability and flexibility. Each DBNAME entry is now defined explicitly by its type (Local, Remote or Host VM (Guest Sharing)). CICS AXE Transaction TPNs (Transaction Program Names) are still included in the directory as a type of ′LOCALAXE′. The DBNAME Directory Builder program, ARICBDID has been rewritten as a REXX/VSE procedure with extensive error and dependency checking. Support for TCP/IP information is added and ′alias′ DBNAMEs are supported. **ALL** DBNAMEs **must** be specified in the new DBNAME Directory, including the Product Default DBNAME ″SQLDS″. A migration REXX/VSE procedure, ARICCDID, is provided to assist in migrating to the new format.

V7R2 and V7R1 Incompatibilities

There are no incompatibilities between DB2 Server for VSE V7R2 and DB2 Server for VSE V7R1.

V7R3 and V7R2 Incompatibilities

There are no incompatibilities between DB2 Server for VSE V7R3 and DB2 Server for VSE V7R2.

Notices

Programming Interface Information

This manual is intended to help system administrators plan and maintain the DB2 Server for VSE database manager.

This manual also documents General-Use Programming Interface and Associated Guidance Information provided by the DB2 Server for VSE database manager.

General-Use programming interfaces allow the customer to write programs that obtain the services of the DB2 Server for VSE database manager.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

General-Use Programming Interface

General-Use Programming Interface and Associated Guidance Information...

End of General-Use Programming Interface

Trademarks

Bibliography

This bibliography lists publications that are referenced in this manual or that may be helpful.

DB2 Server for VSE Publications

- *DB2 Server for VSE & VM Application Programming*, SC09-2889
- *DB2 Server for VSE & VM Database Administration*, SC09-2888
- *DB2 Server for VSE & VM Database Services Utility*, SC09-2983
- *DB2 Server for VSE & VM Diagnosis Guide and Reference*, LC09-2907
- *DB2 Server for VSE & VM Overivew*, GC09-2995
- *DB2 Server for VSE & VM Interactive SQL Guide and Reference*, SC09-2990
- *DB2 Server for VSE & VM Master Index and Glossary*, SC09-2890
- *DB2 Server for VSE Messages and Codes*, GC09-2985
- *DB2 Server for VSE & VM Operation*, SC09-2986
- *DB2 Server for VSE System Administration*, SC09-2981
- *DB2 Server for VSE & VM Performance Tuning Handbook*, GC09-2987
- *DB2 Server for VSE & VM SQL Reference*, SC09-2989

Related Publications

- *DB2 Server for VSE & VM Data Restore*, SC09-2991
- *DRDA: Every Manager's Guide*, GC26-3195
- *IBM SQL Reference, Version 2, Volume 1*, SC26-8416
- *IBM SQL Reference*, SC26-8415

Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA) Publications

- *IBM VSE/ESA Administration*, SC33-6505
- *IBM VSE/ESA Diagnosis Tools*, SC33-6514
- *IBM VSE/ESA General Information*, GC33-6501
- *IBM VSE/ESA Guide for Solving Problems*, SC33-6510
- *IBM VSE/ESA Guide to System Functions*, SC33-6511
- *IBM VSE/ESA Installation*, SC33-6504
- *IBM VSE/ESA Messages & Codes*, SC33-6507
- *IBM VSE/ESA Networking Support*, SC33-6508
- *IBM VSE/ESA Operation*, SC33-6506
- *IBM VSE/ESA Planning*, SC33-6503
- *IBM VSE/ESA System Control Statements*, SC33-6513
- *IBM VSE/ESA System Macros User's Guide*, SC33-6515
- *IBM VSE/ESA System Macros Reference*, SC33-6516
- *IBM VSE/ESA System Utilities*, SC33-6517
- *IBM VSE/ESA Unattended Node Support*, SC33-6512
- *IBM VSE/ESA Using IBM Workstations*, SC33-6509

CICS/VSE Publications

- *CICS/VSE Application Programming Reference*, SC33-0713
- *CICS/VSE Application Programming Guide*, SC33-0712
- *CICS Application Programming Primer (VS COBOL II)*, SC33-0674
- *CICS/VSE CICS-Supplied Transactions*, SC33-0710
- *CICS/VSE Customization Guide*, SC33-0707
- *CICS/VSE Facilities and Planning Guide*, SC33-0718
- *CICS/VSE Intercommunication Guide*, SC33-0701
- *CICS/VSE Performance Guide*, SC33-0703
- *CICS/VSE Problem Determination Guide*, SC33-0716
- *CICS/VSE Recovery and Restart Guide*, SC33-0702
- *CICS/VSE Release Guide*, GC33-1645
- *CICS/VSE Report Controller User's Guide*, SC33-0705
- *CICS Transaction Server for VSE/ESA V1R1.0 Resource Definition Guide*, SC33-0709
- *CICS/VSE Resource Definition (Online)*, SC33-0708
- *CICS/VSE System Definition and Operations Guide*, SC33-0706
- *CICS/VSE System Programming Reference*, SC33-0711
- *CICS/VSE User's Handbook*, SX33-6079

- *CICS/VSE XRF Guide*, SC33-0704

CICS/ESA Publications
- *CICS/ESA General Information*, GC33-0803

VSE/Virtual Storage Access Method (VSE/VSAM) Publications
- *VSE/VSAM Commands and Macros*, SC33-6532
- *VSE/VSAM Introduction*, GC33-6531
- *VSE/VSAM Messages and Codes*, SC24-5146
- *VSE/VSAM Programmer's Reference*, SC33-6535

VSE/Interactive Computing and Control Facility (VSE/ICCF) Publications
- *VSE/ICCF Administration and Operation*, SC33-6562
- *VSE/ICCF Primer*, SC33-6561
- *VSE/ICCF User's Guide*, SC33-6563

VSE/POWER Publications
- *VSE/POWER Administration and Operation*, SC33-6571
- *VSE/POWER Application Programming*, SC33-6574
- *VSE/POWER Networking*, SC33-6573
- *VSE/POWER Remote Job Entry*, SC33-6572

Distributed Relational Database Architecture (DRDA) Library
- *Application Programming Guide*, SC26-4773
- *Architecture Reference*, SC26-4651
- *Connectivity Guide*, SC26-4783
- *DRDA: Every Manager's Guide*, GC26-3195
- *Planning for Distributed Relational Database*, SC26-4650
- *Problem Determination Guide*, SC26-4782

C/370 for VSE Publications
- *IBM C/370 General Information*, GC09-1386
- *IBM C/370 Programming Guide for VSE*, SC09-1399
- *IBM C/370 Installation and Customization Guide for VSE*, GC09-1417
- *IBM C/370 Reference Summary for VSE*, SX09-1246
- *IBM C/370 Diagnosis Guide and Reference for VSE*, LY09-1805

VSE/REXX Publication
- *VSE/REXX Reference*, SC33-6642

Other Distributed Data Publications
- *IBM Distributed Data Management (DDM) Architecture, Architecture Reference, Level 4*, SC21-9526
- *IBM Distributed Data Management (DDM) Architecture, Implementation Programmer's Guide*, SC21-9529
- *VM/Directory Maintenance Licensed Program Specification*, GC20-1836
- *IBM Distributed Relational Database Architecture Reference*, SC26-4651
- *IBM Systems Network Architecture, Format and Protocol Reference*, SC30-3112
- *SNA LU 6.2 Reference: Peer Protocols*, SC31-6808
- *Reference Manual: Architecture Logic for LU Type 6.2*, SC30-3269
- *IBM Systems Network Architecture, Logical Unit 6.2 Reference: Peer Protocols*, SC31-6808
- *Distributed Data Management (DDM) General Information*, GC21-9527

CCSID Publications
- *Character Data Representation Architecture, Executive Overview*, GC09-2207
- *Character Data Representation Architecture Reference and Registry*, SC09-2190

C/370 Publications
- *IBM C/370 Installation and Customization Guide*, GC09-1387
- *IBM C/370 Programming Guide*, SC09-1384

Communication Server for OS/2 Publications
- *Up and Running!*, GC31-8189
- *Network Administration and Subsystem Management Guide*, SC31-8181
- *Command Reference*, SC31-8183
- *Message Reference*, SC31-8185
- *Problem Determination Guide*, SC31-8186

Distributed Database Connection Services (DDCS) Publications
- *DDCS User's Guide for Common Servers*, S20H-4793
- *DDCS for OS/2 Installation and Configuration Guide*, S20H-4795

VTAM Publications
- *VTAM Messages and Codes*, SC31-6493
- *VTAM Network Implementation Guide*, SC31-6494

- *VTAM Operation*, SC31-6495
- *VTAM Programming*, SC31-6496
- *VTAM Programming for LU 6.2*, SC31-6497
- *VTAM Resource Definition Reference*, SC31-6498
- *VTAM Resource Definition Samples*, SC31-6499

CSP/AD and CSP/AE Publications

- *Developing Applications*, SH20-6435
- *CSP/AD and CSP/AE Installation Planning Guide*, GH20-6764
- *Administering CSP/AD and CSP/AE on VM*, SH20-6766
- *Administering CSP/AD and CSP/AE on VSE*, SH20-6767
- *CSP/AD and CSP/AE Planning*, SH20-6770
- *Cross System Product General Information*, GH23-0500

Query Management Facility (QMF) Publications

- *Introducing QMF*, GC27-0714
- *Installing and Managing QMF for VSE*, GC27-0721
- *QMF Reference*, SC27-0715
- *Installing and Managing QMF for VM*, GC27-0720
- *Developing QMF Applications*, SC27-0718
- *QMF Messages and Codes*, GC27-0717
- *Using QMF*, SC27-0716

Query Management Facility (QMF) for Windows Publications

- *Getting Started with QMF for Windows*, SC27-0723
- *Installing and Managing QMF for Windows*, GC27-0722

DL/I DOS/VS Publications

- *DL/I DOS/VS Application Programming*, SH24-5009

COBOL Publications

- *VS COBOL II Migration Guide for VSE*, GC26-3150
- *VS COBOL II Migration Guide for MVS and CMS*, GC26-3151
- *VS COBOL II General Information*, GC26-4042
- *VS COBOL II Language Reference*, GC26-4047
- *VS COBOL II Application Programming Guide*, SC26-4045
- *VS COBOL II Application Programming Debugging*, SC26-4049
- *VS COBOL II Installation and Customization for CMS*, SC26-4213
- *VS COBOL II Installation and Customization for VSE*, SC26-4696
- *VS COBOL II Application Programming Guide for VSE*, SC26-4697

Systems Network Architecture (SNA) Publications

- *SNA Transaction Programmer's Reference Manual for LU Type 6.2*, GC30-3084
- *SNA Format and Protocol Reference: Architecture Logic for LU Type 6.2*, SC30-3269
- *SNA LU 6.2 Reference: Peer Protocols*, SC31-6808
- *SNA Synch Point Services Architecture Reference*, SC31-8134

Miscellaneous Publications

- *IBM 3990 Storage Control Planning, Installation, and Storage Administration Guide*, GA32-0100
- *Dictionary of Computing*, ZC20-1699
- *APL2 Programming: Using Structured Query Language*, SH21-1056
- *ESA/390 Principles of Operation*, SA22-7201

Related Feature Publications

- *DB2 for VSE Control Center Operations Guide*, GC09-2992
- *DB2 Replication Guide and Reference*, SC26-9920

Index

Contacting IBM

Before you contact DB2 customer support, check the product manuals for help with your specific technical problem.

For information or to order any of the DB2 Server for VSE & VM products, contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the U.S.A., then you can call one of the following numbers:
- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product information

DB2 Server for VSE & VM product information is available by telephone or by the World Wide Web at http://www.ibm.com/software/data/db2/vse-vm

This site contains the latest information on the technical library, product manuals, newsgroups, APARs, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:
- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at http://www.ibm.com/planetwide

In some countries, IBM-authorized dealers should contact their dealer support structure for information.

File Number: S370/4300-50
Program Number: 5697-F42

Printed in USA

SC09-2981-03

Spine information:

DB2 Server for VSE System Administration Version 7 Release 5