SPICE

Span Integrated Checkpoint/Restart Environment

SPICE SQL™ Product Description

Release 1.1

SPI 11 05

Span Software Consultants Limited

The Genesis Centre
Birchwood Science Park
WARRINGTON, CHESHIRE WA3 7BH
UNITED KINGDOM

Telephone	+44 (0) 1925 814444
Fax	+44 (0) 1925 837348
email	spice@spansoftware.com
website	www.spansoftware.com

1st October 2002

Preface

Abstract

SPICE is an acronym for SPan Integrated Checkpoint/Restart Environment.

SPICE SQL™ is a software product that simplifies the design, implementation and operation of restartable batch application programs in the DB2 environment.

This document describes SPICE SQL and its principal features. It is intended as an introduction to SPICE SQL.

Other SPICE Manuals

SPI 08 SPICE SQL™ Product Reference Manual
This manual is the principal reference for SPICE SQL, the SPICE product for the DB2 environment.

SPI 09 SPICE DL/I™ Product Reference Manual
This manual is the principal reference for SPICE DL/I, the companion SPICE product for the IMS environment.

SPI 10 SPICE SQL™ & SPICE DL/I™ Diagnostics Description Manual
This manual is the reference for the messages and other diagnostic information issued by the SPICE products.

SPI 13 SPICE SQL™ & SPICE DL/I™ Installation Manual
This manual describes how the SPICE products should be installed.

Note:

Contents

Figures

Chapter 1. Introduction to SPICE SQL™

Using This Manual

This manual is a guide to the function and facilities of SPICE SQL.

This chapter is an introduction to the product. Read this chapter for a brief overview of SPICE SQL and its facilities.

Chapter 2, ***Overview of SPICE SQL Application Programming***, introduces the architecture of SPICE SQL restartable programs, and contains an overview of how SPICE SQL facilities are requested from an application program.

Chapter 3, ***SPICE SQL Operation***, is a summary of the operational responsibilities that SPICE SQL restartable programs require.

Chapter 4, ***SPICE SQL Installation***, discusses the software requirements of SPICE SQL.

Introduction

The explosion in the number of on-line application systems has not removed the need for batch processing. Every industry can quote processes that are most efficiently performed by programs that do not require direct supervision from a terminal. It is a growing trend that commercial and competitive considerations dictate that such processing be performed concurrently with the on-line systems. DB2 applications can be run in such a fashion only if they are restartable.

SPICE SQL is a software package that complements IBM's DB2 system's facilities for batch application programs. It provides a complete environment for the development, maintenance and operation of restartable DB2 application programs.

Users of SPICE SQL benefit from the following:

- **Operational benefits**
  SPICE SQL provides easy to use facilities for controlling SPICE SQL programs. It provides fully automatic restart of failing application programs, simply by resubmission of the job; no JCL changes are required. The same operational technique can be applied to all application jobs using SPICE SQL.
- **Development cost savings**
  The SPICE SQL programming facilities are simple to learn and use. Minimal training of programming staff is required.
  SPICE SQL includes powerful facilities for controlling application program database commit processing. These facilities minimise the programmer's responsibility for commit point frequency. This results in less complex programs.
  Senior development staff are released from the task of developing and maintaining in-house facilities and procedures.
  SPICE SQL results in reduced development and maintenance costs.
- **Application integrity**
  SPICE SQL meets the highest standards of reliability and integrity. The nature and timing of external failures will not compromise the ability of SPICE SQL to restart the application successfully.

SPICE SQL was developed and is marketed by Span Software Consultants Limited. Span Software has been providing software for IBM mainframes since 1976. Furthermore, Span Software has over 10 years of experience in delivering and maintaining software to support restartable applications.

Span Software is proud of its record in providing and maintaining its software worldwide for MVS users in leading companies from all industry sectors.

Restartable Application Programs

Many customers require that increasing numbers of their batch applications co-exist with their on-line applications. The mechanics of Data Base Management Systems require such application programs to be restartable.

The reader will have experienced the frustration that occurs when one is interrupted from reading a book by a telephone call, only to discover upon return that the book has fallen shut. It can take some time to find again one's place in the story. A prudent reader makes intelligent use of a bookmark. In a similar way, restartable programs anticipate the possibility of system failure, and are so written as to minimize the difficulties that will arise when an interruption in processing occurs.

Restartable programs regularly store information defining their current position in both program code and data, typically in databases. Following a failure, they use this data to minimize the processing that has to be repeated. The eventual state of the databases is unaffected by any interruptions that may occur.

Programs that are not restartable have to process again from the beginning, which will often require the recovery of databases from backup copies. This can be a lengthy process. Selecting which backup copies to use can be a complex task, and may be prone to operator error. Furthermore, in the case of programs that update databases alongside the on-line systems, this form of recovery may result in the loss of database updates issued from on-line transactions. This recovery option may be unacceptable to business management, in which case batch applications will have to be restartable.

Implementing restartable programs has previously lead to significant increases in costs, arising in both development and operational areas. SPICE SQL™ and SPICE DL/I™ have been conceived to minimize these additional costs.

SPICE SQL

SPICE SQL is a software product that facilitates the implementation and operation of restartable DB2 application programs. It extends DB2 SQL to provide facilities appropriate to restartable programs, namely management of program data and sequential files required for restart, both synchronized with DB2 commit point processing. SPICE SQL has its own DB2 database for managing its restartable application programs.

SPICE SQL has the following characteristics:

Application Program Interface

- Simple to use facilities.
- Minimal training requirements.
- Minimal understanding of DB2 internals.
- Integration between DB2 tables and OS sequential files.
- Minimal cross-module binding.
- Integration with fourth generation technology, CASE tools etc.

Operating Interface

- Simple to use facilities.
- TSO/ISPF menu system.
- Minimal training requirements.
- Integration with automated operator facilities.
- Fail-safe operational procedures.

SPICE SQL Facilities

SPICE SQL Program Area Management

SPICE SQL program area management allows an application program to inform SPICE SQL as to which of its data areas contain information that the program will require for successful restart. Should the application program fail, it will retrieve this information during restart processing, and restore the program's data areas to their contents at the time of the last commit point prior to the failure.

SPICE SQL Sequential File Management

SPICE SQL includes facilities for the processing of sequential files. SPICE SQL will maintain positioning information, in its database. Should the application program fail, SPICE SQL will retrieve this information during restart processing, and automatically re-position each file to its position at the time of the last commit point prior to the failure.

SPICE SQL Database Commit Point Processing

When SPICE SQL decides to effect a commit point it performs the following:

- Saves the contents of designated application program data areas.
- Saves re-positioning information for sequential files under its control.
- Requests a database commit point by issuing a commit statement.

SPICE SQL Automatic Restart Processing

SPICE SQL automatically detects when an application program is restartable. The same set of JCL can be used for starting, restarting and rerunning SPICE SQL applications, without change. When the program declares its first program area, SPICE SQL will determine if restart is required. When the program executes after a previous failure, SPICE SQL will perform the following restart processing:

- Restore application program data areas upon application request.
- Re-position sequential files.

SPICE SQL facilities are available to override the program restart, so that the program may be rerun.

SPICE SQL In-Flight Restart™

SPICE SQL in-flight restart is an easy to use programming facility that can enable an application program to continue and complete its processing, despite conflicts with other application programs.

If two programs compete for the same database entries the situation can arise where a deadlock occurs. When this happens the database management system will resolve the deadlock by discarding the uncommitted updates of one of them. In certain circumstances the affected application program is informed of this. DB2 informs an application that all uncommitted database updates have been lost with an `SQLCODE` of `-911`. The application program can abandon uncommitted database updates unilaterally, by issuing a rollback statement. In both cases, the non-database files and the program areas are left out of step

with the database contents. Without in-flight restart, the only sensible option available to an application program is that of immediate termination.

SPICE SQL in-flight restart allows the application program to continue and complete its processing without interruption. SPICE SQL will detect the situations described above, and allow the application program to re-commence its processing from the previous commit point by re-issuing restart requests, thereby keeping databases, program areas and sequential files in synchronization.

SPICE SQL Application Testing

The SPICE SQL application testing facility simplifies the development of automated test suites that include thorough testing of application restart logic.

It enables application programmers to imbed breakpoints within SPICE programs. These breakpoints are activated by special DD names in the program JCL. When these DD names are absent, SPICE SQL will ignore the breakpoints. By altering the application JCL, the user can force the application to fail at the breakpoint. The user can then remove the JCL changes, so as to test restart of the program.

SPICE SQL Commit Point Suppression

SPICE SQL can reduce the load on a DB2 system, by minimising the number of commit points. Furthermore, application design can be simplified, by reducing the sensitivity of the application to commit statement frequency.

SPICE SQL monitors the numbers of updated database table rows and the time elapsed, since the previous commit point. SPICE processes database update commit statements. When appropriate, SPICE SQL commit point processing can suppress unnecessary commit points, by returning control to the application, without committing the database updates.

SPICE SQL commit point suppression is fully under the control of the installation. The parameters are stored in a DB2 database. They can easily be modified, by use of the SPICE SQL operational interface, and require no modification to application program code or JCL.

SPICE SQL Application Timeout

SPICE SQL application timeout anticipates problems arising from excessive uncommitted database updates. It can terminate such ill-conditioned programs, before other applications are impacted.

SPICE SQL monitors the number of database rows that have been updated since the previous database commit point. This data is used to control the SPICE SQL facilities of commit point suppression and application timeout.

SPICE SQL application timeout is fully under the control of the installation. The parameters are stored in a DB2 database. They can easily be modified, by use of the SPICE SQL operational interface, and require no modification to application program code or JCL.

Operating SPICE SQL and its Application Programs

SPICE SQL restartable application programs require little intervention from the operator. As SPICE SQL restart requires no JCL change, most application and system failures require only that the program JCL be re-submitted.

Control of the other SPICE facilities is performed with the easy to use SPICE TSO/ISPF operator subsystem.

SPICE SQL and SPICE DL/I

SPICE DL/I™ is a software product that facilitates the implementation and operation of restartable IMS application programs. It enhances IMS symbolic checkpoint/restart, providing significant improvements in application programming costs, operation reliability and ease of use. SPICE DL/I has its own IMS database for managing the restartable application programs.

When both SPICE SQL and SPICE DL/I are installed, the following facilities are available in the IMS environment:

- SPICE facilities are sensitive to both SQL and DL/I activity.
- SPICE SQL applications can be run, unchanged, as IMS BMPs or as IMS batch applications; SPICE supports the SQL COMMIT and ROLLBACK statements from IMS programs.
- SPICE application programs may use either DB2 or IMS databases as their SPICE restart database.

Chapter 2. Overview of SPICE SQL™ Application Programming

This chapter introduces the architecture of SPICE SQL restartable programs, and contains an overview of how SPICE SQL facilities are requested from an application program.

It is divided into the following sections:

SPICE SQL gives the programmer two methods of requesting its facilities. The program can issue requests using the SPICE Restart application program interface (SPICE Restart API). Alternatively, application programs can request SPICE facilities by issuing SQL statements against SPICE SQL restart database tables. Both offer a comprehensive set of facilities for implementing restartable application programs.

SPICE SQL Restartable Program Organization

This section explains the architecture of SPICE SQL restartable programs.

It begins by illustrating the design of SPICE SQL restartable programs with a simple program. It does not initially take into account SPICE sequential files or SPICE commit point suppression. The reader is then walked through the processes of start, restart and rerun.

Re-startable Program Structure

Simple Batch Program

Consider the following figure. It illustrates the structure of a typical batch program.

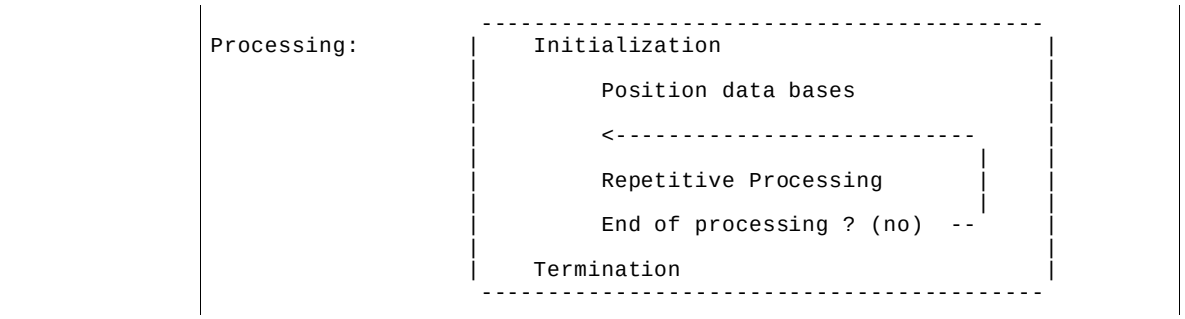

Figure 2.1: Structure of simple batch program

It begins by performing its initialization, which will consist primarily of setting up its data in working storage. It might then open some SQL cursors that it requires. It then proceeds to perform some form of repetitive processing until some criteria are met, when it terminates. It might, for instance, be processing data from a sequential file, in which case it would enter termination processing at end of file.

If this program had to process and update large numbers of database rows, we might experience problems from the large number of uncommitted database entries that the program would hold. To overcome this problem we could decide to add some commit points.

Batch Program with Commit Points

Our program might then look like the following:

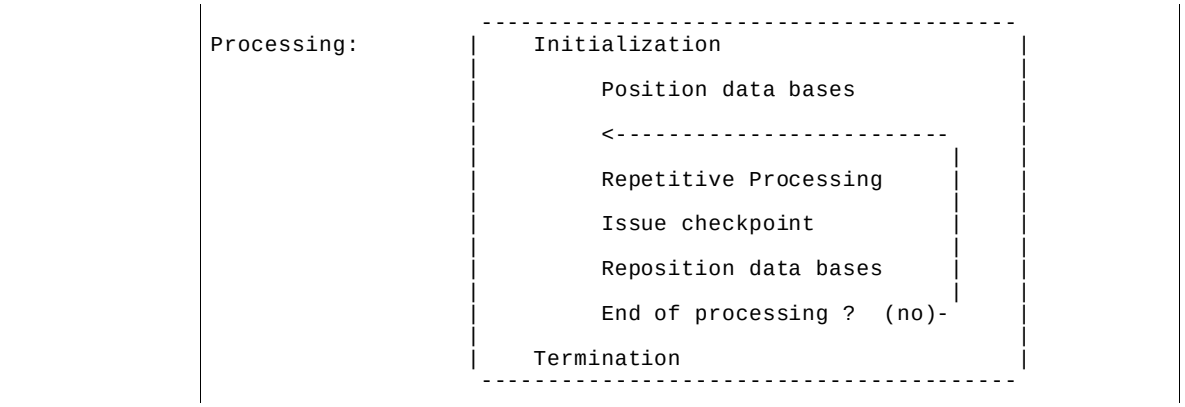

Figure 2.2: Structure of batch program with checkpoints

Our program now issues commit points after processing each iteration of our processing loop. Because DB2, by default, closes any open cursors during commit point processing; we have to re-open them after the commit.

Our program can now execute without holding large numbers of uncommitted updates. A problem might arise, however, should the program, or the system, fail during execution. Our program may have updated, and committed, many database entries. It may prove difficult, or even impossible, to determine which entries have been updated. The conventional strategy with such programs is to recover all the database tables to their state at the beginning of the job, before rerunning the program. This can be a lengthy and complex task. If our database tables have been shared with other updating programs, the on-line network for instance, the option to rerun may not be open to us. Recovering our database tables would undo all updates made by other programs. This may be unacceptable. We may then decide that our program should be restartable.

Batch Program with SPICE SQL

If we use SPICE SQL, our program might then look like the following:

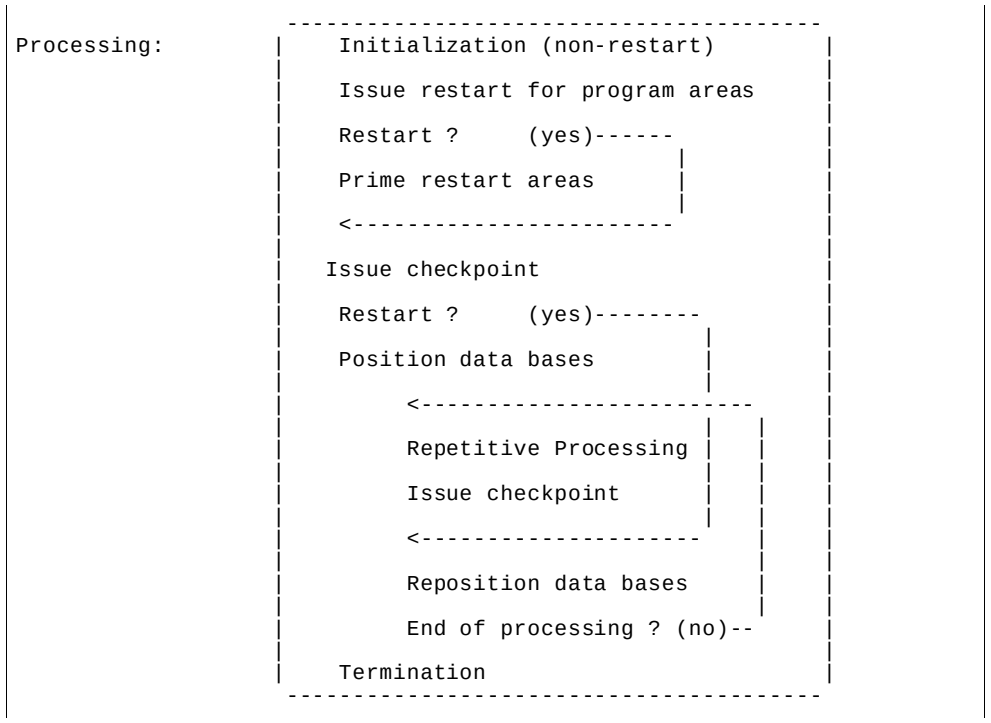

Figure 2.3: Structure of restartable batch program

We have determined those areas of the program required for restart, and issued statements to declare them to SPICE SQL. We have also separated our initialization processing into two parts; that which is required once only, when the program first executes, and that required each time we execute, be it start or restart. Upon restart, we also skip the initial open of the cursors and instead skip forward to the cursor re-open that we perform after each commit point. Our program is now restartable.

Restartable Program Execution

We will now consider some execution scenarios, and walk through the processing that the program we developed in the previous topic will perform.

Program Start

We begin by performing general initialization. We then declare the program areas that we require for restart to SPICE SQL. As this is the start of our processing, SPICE SQL will hold no restart data for us and therefore will respond with the appropriate diagnostic code. We will therefore have to initialize our restart data areas. Typical tasks performed here include obtaining the date, setting up cursor range variables, clearing totals, setting up check sums etc. This is a reasonable moment to issue a commit point. SPICE SQL will now save the initial values of the restart data areas to its database. As we are not restarting, we will now open our SQL cursors before entering our main processing loop.

At the end of each iteration of the program, we issue a commit point. SPICE SQL will save the then current contents of the restart program areas to its restart database. At the end of each iteration of the program, we issue a commit point and re-open our SQL cursors.

If our program runs to completion, SPICE SQL will note the successful termination of the program. SPICE will return no restart data to a subsequent execution of the program.

Program Restart

We will suppose that a failure has occurred whilst our program was executing. Perhaps our program terminated abnormally. Alternatively MVS might have failed while we were running.

We begin, as before, by performing general initialization. We then declare the program areas that we require for restart to SPICE SQL. SPICE SQL will recognise that this is a restart, and restore the restart data from the last successful commit point, to our restart program areas. It will return an appropriate diagnostic code. We must therefore skip that part of our program that performs restart data area initialization. This is again a reasonable moment to issue a commit point. SPICE SQL will now save the current values of the restart data areas to its database.

As we are restarting, we will now skip forward to reopen our SQL cursors before continuing our main processing loop. We will have included cursor control values in our restart data. The cursors will therefore represent the same sets of rows as that which would have been processed had the failure not occurred. Similarly other restart data, such as totals, check sums etc. will be unaffected by the failure.

We can now continue processing, without the failure affecting the processing performed by our program.

Program Rerun

The failure may arise because of a severe program error that has been writing (and committing) incorrect updates to the application tables. Our database tables may be in such a state that the only practical option is to rerun the program. To do this we will have to recover all application data to its state at the beginning of the program's execution. We also have to inform SPICE SQL that rerun is required. We can then start the program again. SPICE will not restore any restart data from its restart tables.

Introduction to SPICE SQL Programming, using the SPICE Restart API

This section contains an overview of the facilities available to SPICE SQL application programs, using the SPICE Restart API.

The SPICE Restart API provides a comprehensive set of facilities for implementing restartable application programs. Application programs request SPICE facilities by issuing statements against the SPICE Restart API module.

SPICE Restart API Statements

Application programs issue requests to the SPICE Restart API by calling subroutine SPIRAPI, specifying a parameter list containing the following:

function code A character string defining the type of request.
SRACA The SPICE Restart API communication area.
other parameters As appropriate to the request.

Language Support

the SPICE Restart API will execute with all programming languages supported by DB2 and IMS.

Program Preparation

The SPICE Restart API high level interface routine must be link-edited with the application program. Include load module SPIRAPI of the SPICE load library.

SPICE must be able to access a SPICE restart database, otherwise it cannot perform a restart. If SPICE is to make use of the SPICE DB2 restart database, the SPICE SQL DBRM must be included in the binding of the DB2 application plan. If SPICE DL/I™ is installed, and SPICE is to make use of a SPICE IMS restart database, the IMS PSB must include a PCB for that database. The PCB need not be included in the PCB list passed by IMS to the program; it can be defined LIST=NO.

If SPICE SQL is to monitor the SQL activity of the application, the SPICE SQL high level interface routine must be link-edited with the application program. For programs that execute under the TSO DSN program, include load module SPIRHLI0 of the SPICE load library. For programs that execute under IMS, include load module SPIDHLI0 of the SPICE load library, a component of the SPICE DL/I™ product.

The SPICE Restart API Communication Area (SRACA)

Every call to the SPICE Restart API must include the location of a SPICE Restart API communication area (SRACA) as the second parameter. The contents of the SRACA are as follows:

COBOL

```
       01  SRACA.
      *   *================================================*
      *   !           S  R  A  C  A                        !
      *   !------------------------------------------------*
      *   !    SPICE  RESTART  API  COMMUNICATION  AREA    !
      *   *================================================*
           05 SRAID           PIC X(08) VALUE 'SRACA   '.
           05 SRALEN          PIC S9(9) COMP VALUE +112.
           05 SRARETV         PIC S9(9) COMP.
           05 SRARETV2        PIC S9(9) COMP.
           05 SRARETV3        PIC S9(9) COMP.
           05 SRARETV4        PIC S9(9) COMP.
           05 SRACODE         PIC S9(9) COMP.
           05 SRASTATE        PIC X(05).
           05 FILLER          PIC X(01).
           05 SRASTAT         PIC X(02).
           05 SRAERRM.
              49  SRAERRML    PIC S9(4) COMP.
              49  SRAERRMC    PIC X(70).
```

Figure 2.4: COBOL definition of SRACA

The program must initialize the SRACA, by assigning the constant "SRACA" to field SRAID, and the length of the communication area to field SRALEN.

The SPICE Restart API supplies INCLUDE members for each of the supported programming languages. They expand into a declaration of the SPICE Restart API communication area.

SPICE Restart API Programming Diagnostics

When the SPICE Restart API detects an error, it inserts a message into the MVS job log. The SPICE Restart API then returns to the program, placing an appropriate value in fields SRACODE, SRASTATE and SRASTAT of the SRACA and an error message in field SRAERRM. The values of SRACODE, SRASTATE and SRAERRM conform to the values of fields SQLCODE, SQLSTATE and SQLERRM of the DB2 SQLCA. The values of SRASTAT conform to the values of the status codes returned by IMS in request PCBs. Upon detecting an error indicator, most application programs enter their standard error handling code, possibly resulting in abnormal termination of the program.

Program Area Management Facilities

Restartable programs define to the SPICE Restart API the data areas that are required for successful restart of the program. SPICE saves the contents of these program areas in its database, for use during restart. Program areas are notified to SPICE by means of statements to the SPICE Restart API module SPIRAPI. Each area is assigned a name by the application program. SPICE maintains distinct sets of program areas in its database for each jobname used by each program. At each commit point, SPICE saves the contents of the declared program areas in its restart database. During restart, SPICE will restore the contents of each program area from its restart database, when they are declared to SPICE.

The following COBOL example illustrates how an area of program storage is declared to the SPICE Restart API as necessary for the restart of the program.

```
    COPY SPIRACAC.
    01 PAM-DECLARE    PIC X(18) VALUE 'PAM-DECLARE'.
. . .
    01 STRUCT-B-ID    PIC X(18) VALUE 'MYMODULE-STRUCT-B'.
    01 STRUCT-B.
       02 NAME          PIC X(32).
       02 AGE           PIC S9(4) COMP.
       02 ADDRESS.
          49 ADDRESSL   PIC S9(4) COMP.
          49 ADDRESSC   PIC X(64).
       02 PHONE-NO      PIC X(24).
       02 STRUCT-B-END  PIC X(4) VALUE '>>>>'.
. . .
    CALL SPIRAPI USING PAM-DECLARE, SRACA,
                       STRUCT-B-ID, STRUCT-B, STRUCT-B-END.
```

COBOL Restart API Example 2.1: Declare data structure necessary for restart

The example declares structure STRUCT-B to the SPICE Restart API as an area of program storage necessary for the restart of the program. It identifies the area with name MYMODULE-STRUCT-B. Any resemblance to the name of the area in the program is coincidental, but convenient.

When the program first executes, no data is restored and an SRACODE of +100 is returned in the SRACA. SPICE also returns an SRASTATE of 02000 and SRASTAT of GE. When the program is restarted after a failure, the SPICE Restart API restores the area contents as saved in the SPICE restart database, and returns an SRACODE of zero. SPICE also returns an SRASTATE of 00000 and SRASTAT of blanks in the SRACA.

Other facilities are available to manipulate program areas, with statements using request codes of SAM-FREEZE and SAM-DELETE. As these facilities are rarely, if ever, required by application programs, we will not discuss them further at this moment.

Sequential File Processing Facilities

The SPICE Restart API provides facilities for the processing of sequential files via the SPICE Restart API statements.

The application program processes SPICE SAM files by issuing statements to the SPICE Restart API module SPIRAPI. The SPICE Restart API accesses the files and manages the data required for restart with its own tables.

Input File Processing

Input files are processed using the SAM-READ request codes. If required, an input file can explicitly be opened or closed by issuing statements using, respectively, the SAM-OPEN or SAM-CLOSE request codes.

The following example illustrates how records are read from SPICE SAM files using SPICE Restart API request code SAM-READ.

```
    COPY SPIRACAC.
    01 SAM-READ    PIC X(18) VALUE 'SAM-READ'.
. . .
    01 INDDN       PIC X(8) VALUE 'INDDN'.
. . .
    01 STRUCT-B.
       02 NAME          PIC X(32).
       02 AGE           PIC S9(4) COMP.
       02 ADDRESS.
          49 ADDRESSL   PIC S9(4) COMP.
          49 ADDRESSC   PIC X(64).
       02 PHONE-NO      PIC X(24).
       02 STRUCT-B-END  PIC X(4) VALUE '>>>>'.
. . .
    CALL 'SPIRAPI' USING SAM-READ, SRACA, INDDN, STRUCT-B.
```

COBOL Restart API Example 2.2: Read record from input file

Records from the file allocated to DD name INDDN are read into I/O area STRUCT-B. The SPICE Restart API returns an SRACODE of zero in the SRACA when each record is successfully read. It also returns an SRASTATE of 00000 and an SRASTAT of blanks. When SPICE reaches end of file, it returns an SRACODE of +100. It also returns an SRASTATE of 02000 and an SRASTAT of GE in the SRACA . When the program first executes, file I/O will begin with the first record in the file. When the program is restarted after a failure, the SPICE Restart API will re-position the file, so that the file I/O will restart from the point of the last successful commit point. In other words, file processing will be unaffected by the interruption.

Output File Processing

Output files are processed using the SAM-WRITE request code. If required, an output file can explicitly be opened or closed by issuing statements using, respectively, the SAM-OPEN or SAM-CLOSE request codes.

The following COBOL example illustrates how records are written to SPICE SAM files.

```
    COPY SPIRACAC.
    01 SAM-WRITE   PIC X(18) VALUE 'SAM-WRITE'.
. . .
    01 OUTDDN      PIC X(8) VALUE 'OUTDDN'.
. . .
    01 STRUCT-B.
       02 NAME          PIC X(32).
       02 AGE           PIC S9(4) COMP.
       02 ADDRESS.
          49 ADDRESSL   PIC S9(4) COMP.
          49 ADDRESSC   PIC X(64).
       02 PHONE-NO      PIC X(24).
       02 STRUCT-B-END  PIC X(4) VALUE '>>>>'.
. . .
    CALL 'SPIRAPI' USING SAM-WRITE, SRACA, OUTDDN, STRUCT-B.
```

COBOL Restart API Example 2.3: Write record to output file

Records from I/O area STRUCT-B are written to the file allocated to DD name OUTDDN. After writing the record successfully, the SPICE Restart API returns an SRACODE of zero in the SRACA. It also returns an SRASTATE of 00000 and SRASTAT of blanks. When the program first executes, file I/O will begin at the beginning of the file. When the program is restarted after a failure, the SPICE Restart API will re-position the file, so that the file I/O will restart from the point of the last successful commit point. In other words, file processing will be unaffected by the interruption.

File Record Length Validation

Programs may request that the SPICE Restart API validates record lengths against the size of the program record I/O area.

Record Length Processing

If required, programs can determine the length of records read from input files and specify the length of records to output files. This facility is principally for use with variable and undefined length record files, but can also be used with fixed length record files.

File Record Position Processing

Programs may request that the SPICE Restart API returns a value representing the current file position. It can also re-position of the file to a point defined by such a value.

Commit Point Facilities

Commit Statements

The SPICE Restart API includes a commit statement. If SPICE SQL is monitoring SQL statements, the program can also use the SQL COMMIT statement. The program must not use SQL COMMIT (or ROLLBACK) statements if SPICE SQL is not monitoring SQL statements. To monitor SQL statements, link edit the application with the SPICE high level language interface routine.

Commit Point Suppression

SPICE commit point suppression is a technique that SPICE employs to control the frequency of application program database commit points. It can thereby reduce the load upon the DBMS. By diminishing the responsibility of application programs for commit point frequency, it can also simplify application design.

The following COBOL example illustrates commit point processing in the SPICE Restart API.

```
    COPY SPIRACAC.
    01 SPICE-COMMIT  PIC X(18) VALUE 'COMMIT'.
. . .
    01 COMMIT-ID     PIC X(8) VALUE 'ID0026'.
. . .
    CALL 'SPIRAPI' USING SPICE-COMMIT, SRACA, COMMIT-ID.
```

COBOL Restart API Example 2.4: Commit point processing

The last parameter, COMMIT-ID, is an optional identifier to the commit point.

After issuing a SPICE Restart API commit statement, the program should examine the SRACA. An SRACODE value of zero indicates that SPICE performed a commit point; the program should re-position its databases and re-open any SQL database cursors. SPICE also returns an SRASTATE of 00000 and an SRASTAT of blanks. An SRACODE value of +100 indicates that the SPICE Restart API suppressed the commit point; database positioning and cursors are unchanged. SPICE also returns an SRASTATE of 02000 and an SRASTAT of GE.

Forced Commit Point

An application program can override SPICE commit point suppression, by utilizing one of the following two techniques.

The application can force a commit point by issuing two commit statements without any intervening processing.

The following COBOL example illustrates this:

```
    COPY SPIRACAC.
    01 SPICE-COMMIT  PIC X(18) VALUE 'COMMIT'.
. . .
    01 FORCE-ID      PIC X(8) VALUE 'FORCE001'.
. . .
    CALL 'SPIRAPI' USING SPICE-COMMIT, SRACA, FORCE-ID.
    CALL 'SPIRAPI' USING SPICE-COMMIT, SRACA, FORCE-ID.
```

COBOL Restart API Example 2.5: Forced commit point processing, using double commit

The last parameter, FORCE-ID, is an optional identifier to the commit point.

An application program can also force a commit point by using the SPICE Restart API request code FORCE-COMMIT.

The following COBOL example illustrates this:.

```
    COPY SPIRACAC.
    01 FORCE-COMMIT  PIC X(18) VALUE 'FORCE-COMMIT'.
. . .
    CALL 'SPIRAPI' USING FORCE-COMMIT, SRACA.
```

COBOL Restart API Example 2.6: Forced commit point processing, using FORCE-COMMIT request code

The request omits to specify an identifier to the commit point.

After the forced commit the program must re-position its databases and re-open any SQL database cursors it is processing.

Application Testing Facilities

The SPICE Restart API includes a facility to assist in the testing of SPICE Restart API restartable programs, the SPICE services test facility. The facility enables economical automated testing procedures to be developed that thoroughly exercise the restart logic of SPICE application programs.

A SPICE services testing statement is imbedded at a point in the program where the programmer wishes to test a failure. The facility is controlled by DD statements in the application JCL. These JCL statements, typically assigned to DUMMY, define what action SPICE is to perform when the application issues a SPICE services test statement, abnormally terminate for instance. When the controlling DD statement is removed, the statement will process successfully. It is thereby possible to design a suite of JCL that thoroughly tests the restart logic of a program.

The application program requests the SPICE services test facility with the SPICE-TEST request code.

The following example illustrates the facility:

```
    COPY SPIRACAC.
    01 SPICE-TEST   PIC X(18) VALUE 'TEST'.
. . .
    01 TESTA  PIC X(8) VALUE 'TESTA...'.
. . .
    CALL 'SPIRAPI' USING SPICE-TEST, SRACA, TESTA.
```

COBOL Restart API Example 2.7: Request the SPICE services test facility

When the statement is executed, SPICE will respond according to the DD names allocated to the job step. If DD name "TESTADIE" is found, SPICE will abnormally terminate the program. If DD name "TESTAERR" is found, SPICE returns an SRACODE of -681 in the SRACA. It also returns an SRASTATE of SR261 and an SRASTAT of ZZ.. The program then typically performs standard error processing. If, however, neither DD name is found, SPICE will return to the application, placing an SRACODE of zero in the SRACA. It also returns an SRASTATE of 00000 and an SRASTAT of blanks.

Introduction to SPICE SQL Programming, using SQL Statements

Application programs can request SPICE facilities by issuing SQL statements against SPICE SQL restart database tables. When the monitoring of SQL statements is enabled, SPICE SQL intercepts all SQL statements issued by the program. It will recognise SPICE SQL requests and perform the requested processing directly; they are not passed on to DB2.

SQL statements issued against application database tables are passed promptly on to DB2 for processing. SPICE SQL monitors the numbers of rows updated and the SQLCODE returned by DB2. Values of SQLCODE that indicate serious problems, notably deadlock conditions, can initiate further SPICE SQL processing. Otherwise, SPICE SQL performs no additional processing.

SQL statements concerned with commit point processing, notably COMMIT and ROLLBACK, initiate additional processing by SPICE SQL.

SPICE SQL Statements

In order for SPICE SQL to distinguish application program SQL statements that request SPICE SQL facilities from SQL statements to application databases, the program must follow certain conventions. SQL statements requesting SPICE SQL facilities must include special host variable values, typically in the WHERE clause. Consider the following COBOL example:

```
    01 SPICE-PAM      PIC X(18) VALUE '!SPICE!_PAM'.
    01 STRUCT-B-N     PIC X(25) VALUE '!AREA!=STRUCT_B'.
    01 STRUCT-B.
       02 NAME        PIC X(32).
       02 AGE         PIC S9(4) COMP.
       02 ADDRESS.
          49 ADDRESSL PIC S9(4) COMP.
          49 ADDRESSC PIC X(64).
       02 PHONE-NO    PIC X(24).
. . .
        EXEC SQL INCLUDE SPIRINCC END-EXEC.
. . .
        EXEC SQL SELECT CHAR_1,INT_1,VARCHAR_1,CHAR_2
            INTO :STRUCT-B
            FROM SPICE_PAM
            WHERE CATEGORY=:SPICE-PAM AND NAME=:STRUCT-B-N
            END-EXEC.
```

COBOL SQL Statement Example 2.8: Declare data structure necessary for restart

The example declares structure STRUCT-B to SPICE SQL as an area of program storage necessary for the restart of the program. It identifies the area with name STRUCT_B.

The column names from table SPICE_PAM are summarised in the following figure:

```
DECLARE SPICE_PAM TABLE

    (CATEGORY   CHAR(18) NOT NULL,
     NAME       CHAR(25) NOT NULL,
     STATUS     CHAR(18),

     CHAR_1  CHAR(254),CHAR_2  CHAR(254),...

     GRAF_1  GRAPHIC(127),GRAF_2  GRAPHIC(127),...

     INT_1  INT,INT_2  INT,INT_3  INT,...

     DEC_1  DEC(15,0),DEC_2  DEC(15,0),DEC_3  DEC(15,0),...

     VARCHAR_1  LONG VARCHAR,VARCHAR_2  LONG VARCHAR,...

     VARGRAF_1  LONG VARGRAPHIC,VARGRAF_2  LONG VARGRAPHIC,... )
```

Figure 2.5: SQL definition of SPICE_PAM table

The column CATEGORY is equated to a host variable containing the SPICE SQL facility identification string. This must take one of the following values:

!SPICE!_PAM	Identifies SPICE SQL statements requesting program area processing.
!SPICE!_SAM	Identifies SPICE SQL statements requesting sequential files processing.
!SPICE!_SERVICES	Identifies SPICE SQL statements requesting SPICE services facilities.

SPICE SQL recognises the statement in the example as a request for SPICE facilities when it observes that the value of 18 character host variable SPICE-PAM begins with the text "!SPICE!". The full value, "!SPICE!_PAM", indicates that the statement is a request for SPICE program area facilities. The statement, being a SELECT, informs SPICE of a program area declaration request.

The host variable equated to column NAME identifies the name that the program has assigned to the area that it is declaring to SPICE SQL. For program area declaration requests, SPICE requires a 25 character host variable, beginning with the text "!AREA!=". The area identification name is extracted from the remaining 18 characters of the host variable, in this case "STRUCT_B". Note that the similarity between the variable name and assigned name is coincidental. Any 18 byte name could be used, "MY AREA" for instance.

The remaining host variables are, by elimination, recognised as the program area. The addresses and lengths of these host variables are passed to SPICE SQL in the parameter list built by the DB2 precompiler. SPICE will consolidate them into the address and length of the program area.

The reader will have noticed the unusual column names specified in the database table declaration, i.e. CHAR_1, INT_1. These *generic* columns are named after the DB2 data types they are defined as: CHAR_1 represents a CHARACTER item, and INT_1 an INTEGER item, for example. Instances of these column names are found in the SELECT statement. They correspond, on a one to one basis, to the variables that constitute the program area named in the INTO clause of the statement: In our example, CHAR_1 matches NAME, CHAR_2 matches PHONE-NO. Note that the SQL statement is processed directly by SPICE SQL; it is never passed to DB2. SPICE SQL processes the host variable or structure; the choice of columns does not affect its processing. These columns exist to satisfy the DB2 precompiler. DB2 requires that a SELECT statement must specify which columns from the referenced table are to be assigned to the host variables of the INTO clause. Furthermore, the type of each selected column must be compatible with the corresponding variable. SPICE SQL is designed to avoid any restrictions on the host variable or host structure defining a program area, or file record area. Any host variable or structure that is acceptable to the DB2 precompiler can be specified. The variety of generic columns in the database ensures that there is a column compatible to DB2. DB2 manual ***DB2 Application Programming and SQL Guide*** contains the definitive precompiler compatibility rules. The following details summarize the principal points:

- CHAR_... columns can be used with any length of character string, up to the DB2 limit of 254 bytes.
- DEC_... columns can be used with any length of packed decimal string, up to the DB2 limit of 15 digits.
- INT_... columns can be used with any length or type of numeric, namely SMALLINT, FLOAT, REAL or DOUBLE PRECISION data types.
- Zoned decimal fields, defined in COBOL as PIC 9(length), are not supported. It is possible to overcome this restriction, by defining a separate structure that re-defines the structure containing the field.
- The DB2 pre-compiler does not support FILLER fields. They should be defined with unique names within the structure, initialized to blanks. Alternatively, a separate structure may be specified that re-defines the structure containing the field.

Language Support

SPICE SQL will execute with all programming languages supported by the DB2 precompiler utility program.

Program Preparation

The SPICE SQL high level interface routine must be link-edited with the application program. For programs that execute under the TSO DSN program, include load module SPIRHLI0 of the SPICE load library. For programs that execute under IMS, include load module SPIDHLI0 of the SPICE load library, a component of the SPICE DL/I™ product.

SPICE must be able to access a SPICE restart database, otherwise it cannot perform a restart. If SPICE is to make use of the SPICE DB2 restart database, the SPICE SQL DBRM must be included in the binding of the DB2 application plan. If SPICE DL/I™ is installed, and SPICE is to make use of a SPICE IMS restart database, the IMS PSB must include a PCB for that database. The PCB need not be included in the PCB list passed by IMS to the program; it can be defined LIST=NO.

Program Area Management Facilities

Restartable programs define to SPICE SQL the data areas that are required for successful restart of the program. SPICE saves the contents of these program areas in its database, for use during restart. Program areas are notified to SPICE by means of SQL statements to the SPICE table SPICE_PAM. Each area is assigned a name by the application program. SPICE maintains distinct sets of program areas in its database for each jobname used by each program. The next figure lists the columns of table SPICE_PAM.

```
DECLARE SPICE_PAM TABLE

    (CATEGORY   CHAR(18) NOT NULL,
     NAME       CHAR(25) NOT NULL,
     STATUS     CHAR(18),

     CHAR_1  CHAR(254),CHAR_2  CHAR(254),...

     GRAF_1  GRAPHIC(127),GRAF_2  GRAPHIC(127),...

     INT_1  INT,INT_2  INT,INT_3  INT,...

     DEC_1  DEC(15,0),DEC_2  DEC(15,0),DEC_3  DEC(15,0),...

     VARCHAR_1  LONG VARCHAR,VARCHAR_2  LONG VARCHAR,...

     VARGRAF_1  LONG VARGRAPHIC,VARGRAF_2  LONG VARGRAPHIC,... )
```

Figure 2.6: SQL definition of SPICE_PAM table

SPICE SQL supplies INCLUDE members for each of the supported programming languages. They expand into SQL declarations of all the SPICE SQL tables, including SPICE_PAM.

Programming Language	COBOL	PL/I	FORTRAN	C
INCLUDE Member Name	SPIRINCC	SPIRINCP	SPIRINCF	SPIRINCD

Table 2.1: SPICE SQL table declaration INCLUDE member names

The application program notifies SPICE of the data areas it requires for restart with SQL SELECT statements against the SPICE table SPICE_PAM. This statement indicates an area whose contents are to be saved during the processing of each COMMIT statement; changes to the contents of the area will be propagated to the SPICE restart database at each COMMIT point. Note that the actual DB2 SPICE_PAM table is never read or written to. For reasons of efficiency and flexibility, SPICE SQL manages the data required for restart with its own tables. It does not pass the statement to DB2.

The following COBOL example illustrates how an area of program storage is declared to SPICE SQL as necessary for the restart of the program.

```
    01 SPICE-PAM      PIC X(18) VALUE '!SPICE!_PAM'.
    01 STRUCT-B-N     PIC X(25) VALUE '!AREA!=STRUCT_B'.
    01 STRUCT-B.
       02 NAME        PIC X(32).
       02 AGE         PIC S9(4) COMP.
       02 ADDRESS.
          49 ADDRESSL PIC S9(4) COMP.
          49 ADDRESSC PIC X(64).
       02 PHONE-NO    PIC X(24).
. . .
        EXEC SQL INCLUDE SPIRINCC END-EXEC.
. . .
        EXEC SQL SELECT CHAR_1,INT_1,VARCHAR_1,CHAR_2
            INTO :STRUCT-B
            FROM SPICE_PAM
            WHERE CATEGORY=:SPICE-PAM AND NAME=:STRUCT-B-N
            END-EXEC.
```

COBOL SQL Statement Example 2.9: Declare data structure necessary for restart

The example declares structure STRUCT-B to SPICE SQL as an area of program storage necessary for the restart of the program. It identifies the area with name STRUCT_B.

When the program first executes, no data is restored and an SQLCODE of +100 is returned for the SELECT statement. When the program is restarted after a failure, SPICE SQL restores the area contents as saved in the SPICE restart database, and returns an SQLCODE of zero.

Other facilities are available to manipulate program areas using SQL UPDATE and DELETE statements against table SPICE_PAM. As these facilities are rarely, if ever, required by application programs, we will not discuss them further at this moment.

Sequential File Processing Facilities

SPICE SQL provides facilities for the processing of sequential files via SPICE SQL statements.

The application program processes SPICE SAM files by issuing SQL statements against the SPICE SQL table SPICE_SAM. SPICE SQL intercepts these statements and performs the appropriate processing. Note that the actual DB2 SPICE_SAM table is never read or written to. SPICE SQL accesses the files and manages the data required for restart with

its own tables. It does not pass the statement to DB2. The next figure lists the columns of table SPICE_SAM.

```
DECLARE SPICE_SAM TABLE

    (CATEGORY     CHAR(18) NOT NULL,
     FILE         CHAR(18) NOT NULL,
     ACTION       CHAR(18),
     RECLEN       CHAR(18),

     CHAR_1  CHAR(254),CHAR_2  CHAR(254),...

     GRAF_1  GRAPHIC(127),GRAF_2  GRAPHIC(127),...

     INT_1  INT,INT_2  INT,INT_3  INT,...

     DEC_1  DEC(15,0),DEC_2  DEC(15,0),DEC_3  DEC(15,0),...

     VARCHAR_1  LONG VARCHAR,VARCHAR_2  LONG VARCHAR,...

     VARGRAF_1  LONG VARGRAPHIC,VARGRAF_2  LONG VARGRAPHIC,... )
```

Figure 2.7: SQL definition of SPICE_SAM table

SPICE SQL supplies INCLUDE members for each of the supported programming languages. They expand into SQL declarations of all the SPICE SQL tables, including SPICE_SAM.

Programming Language	COBOL	PL/I	FORTRAN	C
INCLUDE Member Name	SPIRINCC	SPIRINCP	SPIRINCF	SPIRINCD

Table 2.2: SPICE SQL table declaration INCLUDE member names

Input File Processing

Input files are processed using SQL cursor processing statements. They differ from standard SQL cursor processing in that the cursor is not automatically closed at each commit point.

The following example illustrates how records are read from SPICE SAM files using SQL DECLARE CURSOR, OPEN, FETCH and CLOSE statements against the SPICE_SAM table.

```
    01 SPICE-SAM      PIC X(18) VALUE '!SPICE!_SAM'.
    01 DD-INDDB       PIC X(18) VALUE '!DDNAME!=INDDB'.
    01 STRUCT-B.
       02 NAME        PIC X(32).
       02 AGE         PIC S9(4) COMP.
       02 ADDRESS.
          49 ADDRESSL PIC S9(4) COMP.
          49 ADDRESSC PIC X(64).
       02 PHONE-NO    PIC X(24).
. . .
       EXEC SQL INCLUDE SPIRINCC END-EXEC.
. . .
       EXEC SQL DECLARE INDDB_CURSOR CURSOR FOR
           SELECT CHAR_1,INT_1,VARCHAR_1,CHAR_2
           FROM SPICE_SAM
           WHERE CATEGORY=:SPICE-SAM AND FILE=:DD-INDDB
           END-EXEC.
. . .
       EXEC SQL OPEN INDDB_CURSOR END-EXEC.
. . .
       EXEC SQL FETCH INDDB_CURSOR
           INTO :STRUCT-B
           END-EXEC.
. . .
       EXEC SQL CLOSE INDDB_CURSOR END-EXEC.
```

COBOL SQL Statement Example 2.10: Read record from input file

Records from the file allocated to DD name INDDB are read into I/O area STRUCT-B. SPICE SQL returns an SQLCODE of zero when each record is successfully read. The value +100 is returned when end of file is reached in the file. When the program first executes, file I/O will begin with the first record in the file. When the program is restarted after a failure, SPICE SQL will re-position the file, so that the file I/O will restart from

the point of the last successful commit point. In other words, file processing will be unaffected by the interruption.

Output File Processing

Output files are processed using SQL INSERT statements against table SPICE_SAM. If required, an output file can explicitly be opened or closed by issuing SQL UPDATE statements against the table.

The following COBOL example illustrates how records are written to SPICE SAM files.

```
    01 SPICE-SAM      PIC X(18) VALUE '!SPICE!_SAM'.
    01 ACTION-OPEN    PIC X(18) VALUE '!ACTION!=OPEN_OUT'.
    01 ACTION-CLOSE   PIC X(18) VALUE '!ACTION!=CLOSE'.
    01 OUTDDB-DD      PIC X(18) VALUE '!DDNAME!=OUTDDB'.
    01 STRUCT-R.
       02 FILL-1      PIC X(5) VALUE 'NAME:'.
       02 NAME        PIC X(32).
       02 FILL-2      PIC X(5) VALUE ' AGE '.
       02 AGE         PIC X(3) .
       02 FILL-3      PIC X(9) VALUE ' ADDRESS '.
       02 ADDRESS.
          49 ADDRESSL PIC S9(4) COMP.
          49 ADDRESSC PIC X(64).
       02 FILL-4      PIC X(7) VALUE ' PHONE '.
       02 PHONE-NO    PIC X(24).
. . .
        EXEC SQL INCLUDE SPIRINCC END-EXEC.
. . .
        EXEC SQL UPDATE SPICE_SAM
            SET ACTION=:ACTION-OPEN
            WHERE CATEGORY=:SPICE-SAM AND FILE=:OUTDDB-DD
            END-EXEC.
. . .
        EXEC SQL INSERT INTO SPICE_SAM
            (CATEGORY,FILE,
             CHAR_1,CHAR_2,CHAR_3,CHAR_4,INT_1,CHAR_5,
                VARCHAR_1,CHAR_6,CHAR_7)
            VALUES (:SPICE-SAM,:OUTDDB-DD,:STRUCT-R)
            END-EXEC.
. . .
        EXEC SQL UPDATE SPICE_SAM
            SET ACTION=:ACTION-CLOSE
            WHERE CATEGORY=:SPICE-SAM AND FILE=:OUTDDB-DD
            END-EXEC.
```

COBOL SQL Statement Example 2.11: Write record to output file

Records from I/O area STRUCT-R are written to the file allocated to DD name OUTDDB. SPICE SQL returns an SQLCODE of zero when each record is successfully written. When the program first executes, file I/O will begin at the beginning of the file. When the program is restarted after a failure, SPICE SQL will restart from the point of the last successful commit point. In other words, file processing will be unaffected by the interruption.

File Record Length Validation

Programs may request that SPICE SQL validates the expected file record length against that allocated to the selected DD name, when the file is opened. This is performed by assigning a value to column RECLEN in the open file request. This requires an explicit open file request for output files. Input files are always opened explicitly.

Record Length Processing

If required, programs can determine the length of records read from input files and specify the length of records to output files. This is done by defining the record area as a simple VARCHAR structure. The length field specifies the length of the data field, not including itself. This facility is principally for use with variable and undefined length record files, but can also be used with fixed length record files.

File Record Position Processing

Programs cannot process SPICE SAM file positions with SQL Statements. They may request, however, that the SPICE Restart API return a value representing the current file position. They can also request that the file be re-positioned to a point defined by such a value.

Commit Point Facilities

Commit Point Suppression

SPICE commit point suppression is a technique that SPICE employs to control the frequency of application program DB2 commit points. It can thereby reduce the load upon a DB2 system. By diminishing the responsibility of application programs for commit point frequency, it can also simplify application design.

The following COBOL example illustrates commit point processing in SPICE SQL.

```
        EXEC SQL COMMIT END-EXEC.
    IF SQLCODE = 0
*                   ** OPEN THE CURSORS
        EXEC SQL OPEN DB_CURSOR END-EXEC.
```

COBOL SQL Statement Example 2.12: Commit point processing

After the SQL COMMIT statement the resultant SQLCODE is examined. Failures excepted, it will take the value of either zero or +100. The value zero indicates that a DB2 commit point occurred, in which case the program should re-open any database cursors it is processing. The value +100 indicates that SPICE SQL suppressed the commit point, in which case the database cursors remain open.

Forced Commit Point

An application program can override SPICE commit point suppression, by issuing two SQL COMMIT statements without any intervening processing.

The following COBOL example illustrates forced commit point processing in SPICE SQL.

```
        EXEC SQL COMMIT END-EXEC.
        EXEC SQL COMMIT END-EXEC.
*                   ** OPEN THE CURSORS
        EXEC SQL OPEN DB_CURSOR END-EXEC.
```

COBOL SQL Statement Example 2.13: Forced commit point processing

After the SQL COMMIT statement the program must re-open any database cursors it is processing.

Application Testing Facilities

SPICE SQL includes a facility to assist in the testing of SPICE SQL restartable programs, the SPICE services test facility. The facility enables economical automated testing procedures to be developed that thoroughly exercise the restart logic of SPICE application programs.

A SPICE services testing SQL statement is imbedded at a point in the program where the programmer wishes to test a failure. The facility is controlled by DD statements in the application JCL. These JCL statements, typically assigned to DUMMY, define what action SPICE is to perform when the application issues a SPICE services test statement, abnormally terminate for instance. When the controlling DD statement is removed, the SQL statement will process successfully. It is thereby possible to design a suite of JCL that thoroughly tests the restart logic of a program.

The application program requests the SPICE services test facility by issuing SQL UPDATE statements against the SPICE SQL table SPICE_SERVICES. SPICE SQL intercepts these statements and performs the appropriate processing. Note that the actual DB2 SPICE_SERVICES table is never read or written to. SPICE SQL performs its processing without passing the statement to DB2. The next figure lists the contents of table SPICE_SERVICES.

```
DECLARE SPICE_SERVICES TABLE

    (CATEGORY   CHAR(18) NOT NULL,
     ACTION     CHAR(18) NOT NULL,
     FILE       CHAR(18)          )
```

Figure 2.8: SQL definition of table SPICE_SERVICES

SPICE SQL supplies INCLUDE members for each of the supported programming languages. They expand into SQL declarations of all the SPICE SQL tables, including SPICE_SERVICES.

Programming Language	COBOL	PL/I	FORTRAN	C
INCLUDE Member Name	SPIRINCC	SPIRINCP	SPIRINCF	SPIRINCD

Table 2.3: SPICE SQL table declaration INCLUDE member names

The following example illustrates the facility:

```
    01 SPICE-SERV     PIC X(18) VALUE '!SPICE!_SERVICES'.
    01 ACTION-TEST    PIC X(18) VALUE '!ACTION!=TEST'.
    01 TEST-DD-PREFIX PIC X(18) VALUE 'TESTS'.
. . .
        EXEC SQL INCLUDE SPIRINCC END-EXEC.
. . .
        UPDATE SPICE_SERVICES
            SET ACTION=:ACTION-TEST
            WHERE CATEGORY=:SPICE-SERV
            AND FILE=:TEST-DD-PREFIX
            END-EXEC.
```

COBOL SQL Statement Example 2.14: Request the SPICE services test facility

When the statement is executed, SPICE SQL will respond according to the DD names allocated to the job step. If DD name "TESTSDIE" is found, SPICE will abnormally terminate the program. If DD name "TESTSERR" is found, SPICE will return a non-zero SQLCODE to the program, which will then typically perform standard error processing. If, however, neither DD name is found, SPICE will return to the application, placing an SQLCODE of zero in the SQLCA.

Chapter 3. SPICE SQL™ Operation

Introduction to SPICE Operator Facilities

This section serves as an introduction to the facilities of SPICE SQL that are used for the operational control of SPICE SQL and its applications. Topics covered include the use of operator facilities by installation staff.

Background

SPICE is designed specifically to minimize the involvement of operations staff in the day to day running of restartable SPICE SQL programs. For instance, application program restart requires the operator only to resubmit the job JCL, without modification.

Most installations have, as an objective, the adoption of standard operating procedures across application systems. Reducing the variety and complexity of application operation can both increase system reliability, and reduce operating costs. SPICE SQL can contribute significantly to this objective, as the same simple procedures can be utilized for all application systems that use it.

With the exception of application program restart, SPICE SQL applications are administered either with the SPICE TSO/ISPF operator subsystem or the SPICE utility program. Both provide easy to use facilities that allow the user to report upon and administer applications.

SPICE TSO/ISPF Operator Subsystem

The SPICE operator subsystem is a set of interactive panels and programs that provide an easy to use interface to SPICE. They allow operational personnel to administer SPICE and its applications in an intuitive manner from a TSO terminal, through the use of the subsystem's menus and extensive help and tutorial panels.

SPICE Utility Program

The SPICE utility program is a batch program. It can be used by operational personnel to administer SPICE and its applications. Commands are read from the program's input stream and processed accordingly. Printed reports may be obtained on the contents of a SPICE restart database.

SPICE SQL Operational Responsibilities

This section summarizes the uses to which the SPICE operator facilities can be put by various categories of installation personnel.

Operations Personnel

First Line Operations Personnel

Staff responsible for the day to day operation of SPICE SQL applications can use the SPICE operator facilities for the following:

- To monitor the state of applications, by using the SPICE reporting options.
- Following application program failure, to indicate that the application program is to be rerun, rather than restarted. An installation may require that this decision be made in consultation with support staff.

Operations Support Personnel

Staff responsible for resolving problems in the day to day operation of SPICE SQL applications can use the SPICE operator facilities for the following:

- To determine the state of applications, by using the SPICE reporting options.
- To indicate that the application program is to be rerun, rather than restarted. This decision may require consultation with development staff. Use of this facility can be expected to be infrequent.

Operations Planning Personnel

Staff responsible for planning the implementation of applications that use SPICE SQL can use the SPICE operator facilities for the following:

- To register new application programs in the SPICE restart database.
- To control which SPICE facilities are to be used.

Development Personnel

Staff responsible for developing SPICE SQL applications can use the SPICE operator facilities for the following:

- To monitor the state of their application programs, by using the SPICE reporting options.
- To control which SPICE facilities are to be used, according to the current testing requirements of a program.
- Following application program failure, to indicate that the application program is to be rerun, rather than restarted.

Database Administration and Technical Support Personnel

Staff responsible for the technical aspects of SPICE SQL applications can use the SPICE operator facilities for the following:

- To create and initialize the SPICE restart database.
- To decide which SPICE SQL facilities should be used, and select suitable parameter values for them.

The implementation of these decisions could be performed by operations planning personnel.

SPICE Restart Database Reports

This section describes the reports on the contents of the SPICE restart database that can be obtained from the SPICE operator facilities. They are obtained with the LIST option (SPICE operator subsystem) or LIST command (SPICE utility).

The SPICE restart database contains the following entry types:

Program Entries	Define the SPICE options that are to be used when the named program is executed with SPICE.
Job Entries	Contain, for executing restartable SPICE programs, the information necessary for successful program restart. The data may be spread over more than one entry in the database table. There can be sets of entries for any number of different job names for a program entry.
Default Values Entry	Defines certain SPICE options that are to be used for programs that are designated as using these default values.

Program Entry Reports

The following example is an instance of a formatted report of a program entry.

```
PROGRAM SPIVCOBA

 entry created   91.079 15:54:15.0  last change   91.095 17:40:34.7
 restart entry length   4026

 Commit Point Suppression (DEFAULT VALUES)
      thresholds:                          seconds      updates
                  DB2 & IMS BMP regions      3600        10000
                  IMS BATCH regions             0            0
 Application Timeout (DEFAULT VALUES)
      WTO   upon application timeout
      thresholds:                          seconds      updates
                  DB2 & IMS BMP regions         0            0
                  IMS BATCH regions             0            0
 Commit Point Braking (DEFAULT VALUES)
      wait period:                         seconds
                  DB2 & IMS BMP regions         0
                  IMS BATCH regions             0
```

Report Example 3.1: Formatted Report of Program Entry

PROGRAM	The name of the application program.
entry created, last change	The date and time when the entry was created and last altered by SPICE utility command.
restart entry length	Restart entry size.
Commit Point Suppression	Values of SPICE commit point suppression parameters.
Application Timeout	Values of SPICE application timeout parameters.
Commit Point Braking	Values of SPICE commit point braking parameters.

Other Indicators (when present)

(DEFAULT VALUES)	Indicates that the parameters used to control this feature are taken from the default values entry in the database. Absence of this text indicates that SPICE will take its parameters from the program entry.

Job Entry Reports

The following example is an instance of a formatted report of a job entry.

```
PROGRAM SPIVCOBA

JOB     SPICEJOB
        SPICEJOB step DSNTIRU  program SPIVCOBA
 entry created   91.093 13:43:58.2  last change   91.095 15:38:49.7
 last start      91.095 17:20:50.0
 last restart    00.000  0:00:00.0  last stop     91.095 17:22:21.3
 last commit     91.095 17:20:50.3  identifier  SPI00001
                                                              since last
 Statistics:                                  since job start   start/restart
              commit points issued by program                7              7
              commit points issued to DBMS                   1              1
              number database entries updated                              11
              seconds elapsed                                               2
```

Report Example 3.2: Formatted Report of Job Entry

JOB, step, program — The name of the job, the name of the step in its JCL and the program name.

entry created, last change — The date and time when the entry was created and last altered by SPICE utility command.

last start, last restart, last finish — The date and time when this execution of the program started, was last restarted and completed.
Note: last restart can be zero, indicating that the job has not been restarted since this execution started.
Note: last finish can precede **last start** This indicates that this execution of the program has not yet reached completion; it is either running or has abnormally terminated.

last commit — The time and date when the program last issued a commit point.

identifier — A value that SPICE assigns to each commit point.

commit points ... program — The number of commit points that the program has issued since **last start** and since the latest of **last start** and **last restart**.

commit points ... DBMS — The number of commit points that SPICE has passed on to DB2 since **last start** and since the latest of **last start** and **last restart**. These values will never be higher those of the preceding line in the report.
Note: The two columns only differ after a failure of the program has resulted in a restart, in which case the values in the first column will be higher.

number ... entries updated — The number of table rows that have been updated by the program since the latest of **last start** and **last restart**.

seconds elapsed — The time for which the program has been executing, in seconds, since the latest of **last start** and **last restart**.

Other Indicators (when present)

ACTIVE — This text, when present, indicates that the specified job is executing, or has failed and restart is required. SPICE will perform restart processing the next time the job executes the program. Absence of this text indicates that SPICE will perform rerun processing the next time the job executes the program.

Default Values Entry Report

The following example is an instance of a formatted report of the default values entry.

```
DEFAULT ENTRY

 entry created   90.355 15:04:04.3  last change   91.095 17:17:11.7
 restart entry length   4026

 Commit Point Suppression
      thresholds:                          seconds      updates
                  DB2 & IMS BMP regions      3600        10000
                  IMS BATCH regions             0            0
 Application Timeout
      WTO   upon application timeout
      thresholds:                          seconds      updates
                  DB2 & IMS BMP regions         0            0
                  IMS BATCH regions             0            0
 Commit Point Braking
      wait period:                         seconds
                  DB2 & IMS BMP regions         0
                  IMS BATCH regions             0
```

Report Example 3.3: Formatted Report of Default Values entry

DEFAULT ENTRY	Indicates that this is the default values entry.
entry created	The date and time when the database was initialized.
last change	The date and time when the entry was last altered by SPICE utility command.
restart entry length	Restart entry size.
Commit Point Suppression	Values of SPICE commit point suppression parameters used by programs whose database entries specify use of the default values.
Application Timeout	Values of SPICE application timeout parameters used by programs whose database entries specify use of the default values.
Commit Point Braking	Values of SPICE commit point braking parameters used by programs whose database entries specify use of the default values.

Chapter 4. SPICE SQL™ Installation

This chapter discusses the software requirements of SPICE SQL.

Requirements

All SPICE products will execute under any release of MVS/XA and MVS/ESA supported by IBM. The SPICE operator subsystem will execute under any release of TSO/E and ISPF supported by IBM.

SPICE SQL will execute under any release of DB2 supported by IBM.

SPICE Installation

SPICE SQL is supplied on a standard labelled 3480 magnetic tape cartridge, accompanied by documentation detailing the tape format and contents.

It is designed to be installed into its own SMP/E global, DLIB and target CSIs. A complete set of installation jobs are provided.

SPICE SQL includes a sample application. This set of programs and databases verify the successful installation of SPICE SQL.

Glossary

APAR. Authorized Program Analysis Report. A type of SMP/E sysmod.

AIB. The IMS application interface block. A component of IMS. The AIB interface allows DL/I programs to request IMS facilities independently of the PCB list.

application timeout. A SPICE technique for detecting application programs that have not issued a commit point.

bind. The DB2 process whereby DBRMs created by the DB2 precompiler are bound together with the database definitions to create an application plan.

BMP. Batch Message Program. An IMS batch program that executes under the control of an IMS DBCTL.

CAF. Call Attachment Facility. An interface program between an application program and the DB2 system.

CASE. Computer Assisted Systems Engineering.

CICS. Customer Information Control System. An IBM transaction manager.

column. The vertical component of an SQL database table.

commit. The process whereby database updates are applied to databases. Until a commit point is reached, DB2 will prevent other programs from accessing database elements against which uncommitted updates exist.

commit point braking. A SPICE technique for reducing the rate at which an application program consumes resources.

commit point suppression. A SPICE technique for controlling the frequency of commit points from application program programs.

CSI. SMP/E Consolidated Software Inventory.

cursor. A named pointer into an SQL database table subset, defined by program specified selection criteria.

database. For DB2, a group of SQL tables. For IMS, a set of segments, structured hierarchically.

DB2. IBM Database 2. An IBM database management system.

DBCTL. IMS DB control region.

DBMS. Database Management System, i.e. DB2 or IMS.

DBRM. Database Request Module. An object created by the DB2 precompiler. DBRMs are bound together to create an application plan.

DD name. The label assigned to a JCL DD statement.

DD statement. An MVS JCL dataset definition statement.

DL/I. Data Language/One. The query language for IMS databases.

DLIB. SMP/E Distribution Library.

Dynamic SQL. Access requests to SQL databases that are created whilst an application program is being executed.

ESA. Enterprise Systems Architecture. MVS/ESA is an IBM operating system.

forced commit point. A SPICE technique for ensuring an immediate commit point, even if commit point suppression is active.

hexadecimal dump. A report of the contents of an area of storage, presented in hexadecimal digits (0-9,A-F).

HLI. High level Language Interface module. The module that conveys program requests to the DBMS.

host structure. A structure of program variables referenced by an SQL statement.

host variable. A program variable referenced by an SQL statement.

IMS. Information Management System. An IBM database and transaction management system.

IMS DB. IMS database management system, a component of IMS.

IMS TM. IMS transaction management system, a component of IMS.

in-flight restart. A SPICE technique that enables an application program to continue processing after deadlock or program initiated failure, by re-issuing its restart requests.

INCLUDE members. A facility that allows a program to include text from a member of a separate library.

ISPF. The Interactive System Productivity Facility program product. It provides services for supporting on-line panel and menu driven applications under the TSO/E program product.

JCL. MVS Job Control Language.

JES. Job Entry System. The MVS job management system.

MVS job log. The report of a program's execution produced by MVS.

MVS. Multiple Virtual Storage. An IBM operating system.

OS sequential files. Disk or tape files, that are processed in physical sequence.

plan. A DB2 object that represents the database access requirements of an application program. It is created by the DB2 utility BIND command from the DBRMs of the modules of the program.

precompiler. The DB2 utility program that converts the SQL statements in an application program into a form that can be processed by the language compiler. It also creates the DBRM for the module.

PTF. Program Temporary Fix. A type of SMP/E sysmod.

reposition. The process of moving the current record pointer in a sequential file.

rerun. Execution of an application program after a failure, where all processing is repeated.

restart. Execution of an application program after a failure, where only processing performed after the last commit point is repeated.

Restart API. The SPICE restart application program interface. A component of SPICE SQL. The Restart API is used by application programs to request SPICE facilities.

restart database. The database where SPICE keeps all its information relating to its restartable programs.

rollback. The process by which an application program's uncommitted database updates are abandoned.

row. The horizontal component of an SQL database table.

segment. A record in an IMS database.

SMP/E. System Modification Program/Extended. A component of MVS that is used to install and maintain system software.

SPICE_PAM. The table against which SPICE SQL programs request SPICE program area management facilities.

SPICE_SAM. The table against which SPICE SQL programs request SPICE sequential file facilities.

SPICE_SERVICES. The table against which SPICE SQL programs request SPICE services facilities.

SQL. Structured Query Language. The query language for DB2 databases.

SQLCA. SQL Communication Area. A control block that serves as interface between an application program and DB2.

SQLCODE. A field in the SQLCA control block.

SQLERRM. A field in the SQLCA control block.

symbolic checkpoint/restart. An IMS facility for implementing restartable application programs.

sysmod. SMP/E System Modification.

table. A component in an SQL database. Tables consist of columns and rows.

TSO DSN. The DB2 TSO command processor. In particular, this program can attach DB2 application programs.

TSO/E. Time Sharing Option/Extended. A component of MVS.

VARCHAR. Variable length character string. An SQL data type.

VSAM. Virtual Storage Access System.

XA. Extended Architecture. MVS/XA is an IBM operating system.

Index

SPICE SQL™

Product Description Manual

Release 1.1

SPI 11 05

Reader's Comment Form

This manual is published by:

Span Software Consultants Limited
The Genesis Centre
Birchwood Science Park
WARRINGTON
CHESHIRE WA3 7BH
GREAT BRITAIN

Telephone +44 (0) 1925 814444
Fax +44 (0) 1925 837348
Email spice@spansoftware.com
Website www.spansoftware.com

Span Software Consultants Limited welcomes comments on its publications. You are invited to use this form to communicate your comments about this publication, its organization or subject matter.

If you have applied any technical newsletters (TNLs) to this manual, please list them here:

Comments (please include specific chapter and page numbers)

If you want a reply, please provide the following information:

Name _____ Date _____

Company _____ Phone No. _____

Address _____

Thank you for your cooperation.

1 October 2002

SPICE SQL™
Product Description Manual

Release 1.1

SPI 11 05

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

Span Software Consultants Limited
The Genesis Centre
Birchwood Science Park
WARRINGTON
CHESHIRE WA3 7BH
GREAT BRITAIN

Fold and tape | Please do not staple | Fold and tape

1 October 2002