



VS/2

VSAM to DB2 MIGRATION

Technical Overview

Circle Computer Group Ltd.
Harrow House, 23 West Street Haslemere, Surrey, GU27 2AB, UK
Tel: +44 (0)1428 664500 Fax: +44 (0)1428 664509
www.circle-group.com

VS/2 Technical Overview

Contents

What is VS/2?	1
What are the main advantages of using VS/2?	2
Does VS/2 offer any other benefits?	3
What effect does VS/2 have on performance?	4
Does VS/2 support all VSAM data set functions?	5
Can you provide practical help to evaluate VS/2?	6
What does VS/2 consist of?	7
How should I start using VS/2?	8
How are applications maintained after migration?	9
What is the typical VS/2 migration process?	10
Step 1 - Decide migration unit	10
Step 2 - File analysis.....	10
Step 3 - Create DB2 table(s).....	11
Step 4 - Map file(s)	11
Combining steps 2, 3 and 4.....	11
Step 5 - Migrate data.....	12
Step 6 - Test mapping	12
Step 7 - Test application programs.....	12
Step 8 - Production cutover.....	13
Re-engineering your data	14
Field Build Exits – enhancing data.....	14
Field Build Exits – data verification	15
Redefined file structures	15
The primary table in VS/2.....	16
Option 1 - Single Table with no column-level granularity.....	16
Option 2 - single table with column-level granularity	17
Option 3 - additional tables for each variable copybook layout.....	18
VS/2 main menu	20
A typical VS/2 migration project team.....	23
Preparing your applications for VS/2.....	24
Accessing migrated data sets from batch programs	24
Accessing migrated data sets from CICS programs.....	24
VS/2 debugging and trace facilities.....	26
Interpreting bad SQL codes	26
VS/2 trace facility	26
Activating the trace for batch.....	27
Activating the trace in CICS.....	28
Installing VS/2.....	29
Download installation tape.....	29
Run the customisation application	30
Create DB2 tables for the VS/2 dialogs	30
Bind VS/2 DBRMs.....	30
Make VS/2 dialogs available through ISPF	30
Define VS/2 resources to CICS.....	30
Install the VS/2 subsystem	31
Run the VS/2 IVP	31

IBM, ISPF, IMS, CICS, DB2 and MVS are trademarks, registered trademarks, or products of International Business Machines Corporation in the United States, other countries, or both.

VS/2 and DL/2 are registered trademarks of CIRCLE Computer Group, Ltd.

This document applies to VS/2 V1.2.

What is VS/2?

There are many software products available to migrate data from VSAM to DB2. The problem with them is that this still leaves you to undertake the most expensive part of the migration; the manual rewrite of your application programs to access the data in its new form.

VS/2 goes much further. Not only is it a VSAM to DB2 data migration tool; it also allows you to continue to run your existing VSAM application programs, and best of all, *you do not need to change a single line of code.*

It is a complete solution for migrating VSAM data sets to DB2. When your existing batch and online programs run, it automatically intercepts calls to VSAM data sets and redirects them to DB2. The data retrieved from DB2 is reformatted and returned to your application programs in exactly the same format produced by VSAM. Similarly, update calls issued by your programs are converted into the equivalent DB2 SQL call. In addition, VS/2 builds and returns the appropriate VSAM return and reason codes, so that the logic in your programs that is dependent on current calls to VSAM files is preserved.

And most importantly, VS/2 is able to do this without any changes required to your existing application programs.

VS/2 is the sister product of DL/2, which was first installed in a production environment in the mid-1990s. Since then, many hundreds of IMS databases have been successfully migrated to DB2, and many millions of CICS and IMS/DC transactions continue to access them. The specialist programming techniques from DL/2 have found their way into VS/2, so although it may seem like a new product to you, its foundation is tried and tested.

Circle Computer Group has recognised experts in the business of data and database migration. Therefore, VS/2 is not only a unique, but is also **a complete set** of all the elements you need to migrate your VSAM files to DB2 in terms of product functionality and expertise.

This publication is intended for database administrators and systems programmers who will be responsible for implementing VS/2. It answers questions that are frequently asked by potential users of VS/2; it explains the steps required to migrate VSAM data sets to DB2 with VS/2; and it explains how to install VS/2.

What are the main advantages of using VS/2?

VS/2 gives you several very important advantages when you migrate data from VSAM to DB2. Here are some technical benefits;

- ❑ You migrate your data on a file-by-file basis. Thus you can perform the migration in a steady, controlled manner, as resources allow. Within a single application program, you can have a mixture of calls to migrated files and to non-migrated files.
- ❑ Migration is a one-time activity and in many cases is handled by VS/2 automation. Once a file has been converted, VS/2 manages ongoing data access from your existing application programs. At the same time, you can access the data in the DB2 tables using SQL. This can be static SQL, either from new programs or code added to existing programs or dynamic SQL from end-users. Once the data is in DB2 you decide how it is accessed.
- ❑ The working data exists in only one place. You do not need to be concerned with the risks of losing data integrity, the effort of maintaining data synchronisation, or the inefficiencies of data replication and redundancy.
- ❑ You do not have to change your existing applications. Your investment in those applications is fully protected.
- ❑ And finally, all of the SQL that accesses the migrated data sets is static.

It is difficult to overstate the benefits that are offered by these fundamental advantages, which are inherent to the design of VS/2. But more importantly, with your data in DB2 there is any number of potential business benefits.

Does VS/2 offer any other benefits?

Some VSAM files will contain multiple record types, repeating group fields and large redefined areas. VS/2 provides you with APIs so that you can separate data from a single file into multiple DB2 tables. For example, if you have a file that contains a processing control record, you can migrate the control record to one DB2 table, and the remaining data records to a second DB2 table.

Our experience suggests that most VSAM files contain single record types, and these will be converted to single DB2 tables. Assuming that the DB2 table design will be based on your file copybooks, VS/2 can automatically generate the DB2 DDL, the VS/2 mapping data and run time driver modules, saving considerable time and effort.

Another feature of VS/2 is the support it provides for extensive **re-engineering** of your databases during the migration process. Thus you can fully exploit the potential of DB2; you are not limited by any restrictions imposed on your existing design.

Data re-engineering varies in complexity. At a simple level, you can store date fields in DB2 DATE columns, knowing that VS/2 will handle the data translation automatically. This allows you to fully exploit the time series processing capabilities of DB2.

When deciding how much re-engineering to do, you should keep in mind that, as with any database design process, re-engineering is a trade-off between the theoretical benefits and the practical implications. So, just as most organisations will back away from implementing a fully normalised relational database design because of performance reasons, similar practical considerations will come into play when determining the extent of database re-engineering with VS/2.

What effect does VS/2 have on performance?

There are three aspects to the CPU overhead associated with VS/2. These are;

- ❑ There is an inevitable path-length increase due to VS/2. Because VS/2 is built on the tried and trusted architecture of DL/2, its sister product from Circle Computer Group, the path-length increase is minimised, and in most cases will be less than 8%.
- ❑ The cost of processing data in DB2 is greater than the cost of using native VSAM access. However, DB2's ability to manage effective usage of virtual storage for processing large data volumes helps to offset the CPU increase.
- ❑ There is a cost associated with data re-engineering. One reason for this is that the DB2 cost of processing DB2 DATE and DECIMAL columns is greater than the cost of processing CHAR columns. This is because DB2 has to handle the data transformation from its internal representation into the form it returns to the application.

As you would expect, there is no simple answer to the questions of performance and costs. Based on actual customer experiences with DL/2, transaction response times are comparable with no discernible increase. For batch programs, the CPU costs are always increased, although the actual elapsed times can be reduced in some cases.

Does VS/2 support all VSAM data set functions?

VS/2 supports all of the different VSAM processing modes; sequential, direct and skip-sequential. Read backwards is supported as well as multiple, independent reads (such as EXEC CICS BROWSE). VS/2 also provides support for commonly used Access Method Services macros, such as SHOWCB.

VS/2 supports fixed and varying length key sequenced data sets (KSDS), and fixed and varying length relative record data sets (RRDS). Access via an alternate index path is supported.

In the current release, the following are *not* supported

- ❑ Access to an AIX cluster as a regular data set
- ❑ Entry sequenced data set types (ESDS)
- ❑ Liner data set types (LDS)
- ❑ Processing using RBA access
- ❑ Asynchronous VSAM requests

VS/2 supports application programs written in Assembler, PLI and Cobol.

Can you provide practical help to evaluate VS/2?

Circle Computer group has many years' experience in the data migration business; existing DL/2 customers across the world will bear testimony to this. We are familiar with aspects such as data re-engineering and multiple redefined records, and a major part of the VS/2 development focus is on these areas.

The experiences we have gained with DL/2 have been key factors in the design of VS/2 as a toolset: we have tried to develop automated solutions to tasks that were previously undertaken manually.

We know that migrating a file consists of a number of stages, and each stage is supported by a component of VS/2. This is illustrated in the table below;

MIGRATION STAGE	VS/2 TOOL USED
Analysis of the File Characteristics	VS/2 manual and automated mapping component
Creation of the DDL	
Mapping of Alternate Indexes	Batch AIX mapper
Unloading the existing data from VSAM	VS/2 UNLOAD utility
Converting the data into DB2 format	VS/2 Conversion utility
Application testing	Dual Mode facility

The combination of the complete VS/2 toolset, coupled with the data migration knowledge and experience of Circle Computer group and its partners will help to ensure a successful migration.

And if you want to try it out yourself first, VS/2 comes with a 30 -day free trial. Alternatively, you could perform a pilot test, with onsite support from Circle if required.

What does VS/2 consist of?

The VS/2 components to help you with the initial data set migration are shown on the previous page. This part of VS/2 is referred to as the **Data Migration Component**.

There are two other components of VS/2. The first deals with the interface between your application program and VS/2. For batch programs, this is performed using an MVS subsystem that uses DB2 Call Attach facility to issue SQL. In CICS, the interception is done by a VS/2 Global user exit (GLUE).

The call interception for CICS and Batch programs is referred to as the **VS/2 Run-Time Component**. At run-time, VS/2 uses two driver modules for each migrated VSAM data set. These are;

Data set Information Module (DIM)	This defines the relationship between the VSAM record structure and the DB2 table it relates to
Data set Driver Module (DDM)	This contains the necessary SQL to access the DB2 table

These modules are generated by standard VS/2 ISPF dialogue functions. This is the third of the main VS/2 components, and known as the **VS/2 Mapping Component**.

When you purchase VS/2, all of the components and all of the utilities within these components are included.

VS/2 Version 1.2.0 introduces a new component called the **Dual Mode Facility (DMF)**. This component helps you with application testing. In fact, your entire application testing process is effectively automated. There is more detail about this function later.

It works by processing each VSAM call from your application programs twice; once against the VSAM file and once against the data migrated to DB2. DMF automatically compares the data returned, and the return and reason codes (and RESP/RESP2 codes in CICS). If a difference is detected, diagnostics are gathered and the application program is terminated.

The facility is especially useful if you are using VS/2 exits to restructure or re-engineer one or more of your VSAM data sets.

How should I start using VS/2?

A good start is to migrate a single, low-visibility data set. This allows you to explore the capabilities of VS/2 and become familiar with the technology, in terms of both its functionality and how it integrates within your existing environment. As your VS/2 experience and knowledge develops, you can make better decisions to improve the value of the migrated DB2 data, and improve the entire migration process.

A key aspect of VS/2 is the data re-engineering capabilities provided through user APIs. For example, if you have multiple copybooks for the same data set, and the respective record structures differ, you may want to separate this single data set into multiple DB2 tables. This capability is available through VS/2 APIs, which provides support for two types of user exit programs;

- Field Build Exits (FBEs)
- Insert, Replace Delete Exits (IRDs).

Once you have sufficient VS/2 knowledge, the experience of testing migrated data sets, you should consider using VS/2 exits. It is very likely that another VS/2 site (or DL/2 site) has had a similar requirement to yours, in which case it may be possible to exchange experiences (and maybe even some VS/2 user exit source code!).

There is more detail later in this document regarding VS/2 exits and data re-engineering.

How are applications maintained after migration?

Periodically, you may need to change application programs that access VS/2 migrated files. VS/2 does not impact the way you manage application program changes. In fact, with the data in DB2, application changes may be simplified because you can issue new database calls in SQL. VS/2 will work quite happily with programs that issue SQL natively.

You might decide to add a new column to a VS/2 migrated DB2 table. For example, a banking application may want to add a new column to display data in a different currency. Once again, this is no problem for VS/2.

In fact, the only time where VS/2 is significant is when you make a change affecting either an existing call to a VS/2 migrated VSAM file, or when if you add SQL to a program *for the 1st time*. (you need to change your linkage-edit procedures in this case)

Our experience with DL/2 customers is that the number of application program changes reduces after the data is in DB2. This is because of ease of access to DB2 data, coupled with the availability of many ad hoc SQL End User query tools.

What is the typical VS/2 migration process?

Earlier, we looked at the different VS/2 utilities and where they are used in the migration process. A more detailed walk-through of the typical migration process is split into 8 steps.

Step 1 - Decide migration unit

This first step is to decide which file or files you want to migrate at the same time, referred to as a Migration Unit. VS/2 does not impose any rules; make your decision based on factors such as the criticality of the file, its size, the required availability and so on. You could also decide to migrate the file at the same time as some other routine application changes, to reduce application testing.

Step 2 - File analysis

Next, you should analyse the file or files you are going to migrate. There are a number of reasons for doing this.

- (a) You must identify all of the application programs that access the file. This allows you to develop a test plan, and identify the batch procedures that need to be modified to enable VS/2 access and the CICS programs/txns to test.
- (b) Some of the fields in the file, and even some of the records in the file may be obsolete. You should choose whether you want to migrate potentially obsolete data to DB2, or take the opportunity to clean up the file.
- (c) You should understand what future requirements exist for the file, once it has been migrated to DB2. For example, you might want to add new columns to the table to support enhancements that you plan to make to the application.
- (d) Some files may have multiple copybooks. You need to decide how you will handle this.
- (e) You must establish the characteristics of the path to identify all alternate indexes. You will need to create a DB2 index to support all alternate index paths.

When you conclude this phase, you will know which files you are migrating, the DB2 objects that are required, and the basic design of the DB2 tables that will be used.

Step 3 - Create DB2 table(s)

The DDL for the DB2 tables and indexes is produced in this step. In most cases, your DB2 table will look similar to your file copybook, based on the outcome of the File Analysis step.

However, there may be characteristics of the file that may influence your ultimate table layout. For example, it is good DB2 DBA practice to group together those columns that are most frequently updated (to optimise DB2 performance and logging).

The relative position of a field in your copybook does not dictate the position of the column within the DB2 table. You can arrange the columns in your DB2 table to optimise DB2.

Step 4 - Map file(s)

The mapping process establishes the relationship between the fields in the file and the appropriate columns in the DB2 table. Obsolete fields or FILLER fields do not need to be mapped to a DB2 column; these fields will be initialised to a user-defined default value by VS/2.

When you have completed the mapping for a file, you generate the VS/2 runtime driver modules. These are the Data set Information Module (DIM), and the Data set Driver Module (DDM).

Combining steps 2, 3 and 4

The mapping process can be either manual or automated. The actions in steps 2, 3 and 4 assume the file is manually mapped. These can be combined if you use the automated mapping facility. You can use this when you are migrating a VSAM file to a single DB 2 table. The automated mapping facility performs the following tasks:

- (a) Automatic analysis of the VSAM cluster to determine size of the DB2 tablespace and indexspace objects. It also identifies all alternate indexes defined for the file.
- (b) Automatic generation of DDL for tables, indexes and tablespaces
- (c) Automated mapping of the base cluster plus all alternate indexes
- (d) Automatic generation and submission of the jobs to produce the VS/2driver modules.

Step 5 - Migrate data

Data Migration uses a combination of VS/2 and DB2 utilities, and is a 3-step process.

- (a) The VSAM file is unloaded to a sequential data set using the VS/2 VIDUNLOD program
- (b) The sequential data set is converted into a DB2 LOAD format data set using the VS/2 VIDLOAD program. VS/2 services are invoked at this stage to handle any field re-engineering, depending on how the file has been mapped
- (c) The DB2 LOAD utility loads the output data set from step (b) into the DB2 table.

This migration process can involve some degree of data cleansing, particularly where you are mapping fields to DECIMAL or DATE column types. You may discover that your data is not in such a good a state as you thought.

Step 6 - Test mapping

Before you start to test your application programs, verify that the mapping is correct. This is especially important if you are performing and data re-engineering.

The method for this uses the VIDUNLD program, which you used in step 5(a) to unload the VSAM file. Use this in conjunction with the dual mode facility to automatically compare the data in VSAM and DB2.

Step 7 - Test application programs

This is likely to be the longest of the phases in your migration cycle. Normally, it is not necessary to test every program that accesses the file you have migrated, but this depends on the criticality of the file.

As a rule of thumb, the amount of testing you perform should be a factor of the amount of re-engineering of your file.

Using DMF achieves a significant degree of automation in your application testing. DMF requires that you have your data synchronised in VSAM and DB2. When you run your application programs with DMF, VS/2 automatically compares data areas and return codes to ensure that they match exactly. If a difference is encountered, appropriate diagnostics are provided to assist problem determination.

To enable DMF for batch program testing, you specify an additional parameter in the SUBSYS statement for your migrated data set(s). You also supply an additional DD statement for the VSAM file. In CICS, you use VS/2 transactions to start and stop DMF, and you define a new file to CICS. You can also activate DMF in the PLT so that DMF persists across multiple invocations of CICS.

Step 8 - Production cutover

The file or files you are converting must be made unavailable to your application programs until the migration is completed and you have tested VS/2 access to your satisfaction. The time for this may range from a few minutes to many hours.

In practise, there are a number of production cutover tasks that can be performed in advance of the cutover, to reduce the period when the file is unavailable. Ideally, the only tasks you want to perform are to convert the data and test the migration. Creation of the DB2 objects, binding of the DB2 packages and so on should be done some time before the actual cutover.

Re-engineering your data

The ability to re-engineer is a common requirement for data migration tools. As was said earlier, VS/2 has data re-engineering capability at several levels;

- (a) At a field level. For example, you can store Zoned decimal data in a DECIMAL DB2 column. This allows new programs or end-users to exploit the arithmetic processing capabilities in DB2. This re-engineering is performed automatically when you decide to use DECIMAL columns with Zoned Decimal fields during the initial data set mapping step.
- (b) Another example of field level re-engineering is date fields, which you can store in DB2 DATE columns. All you have to do is to describe the format of the date field in the initial data set mapping, and VS/2 does the rest. Once again, this opens up many possibilities for processing the data using DB2 built-in functions.

The typical field-level data re-engineering can be performed by VS/2 without the use of any user-written code. However, the API in VS/2 for FBEs and IRDs mean that you can significantly extend the re-engineering possibilities. Typical uses of FBE and IRD exits follow;

Field Build Exits – enhancing data

Assume that you have an application that uses BIT fields for application or data flags. Suppose you have an 8-BIT field value of B"01100001". This could be stored in a CHAR(1) DB2 column, and the value would be X"61", which translates to the character "/". This value is probably not meaningful to DB2 query users.

To make the data meaningful, you can develop a VS/2 Field Build Exit so that the BIT data is translated into character data. This can be stored in 8 CHAR(1) DB2 columns with meaningful names, such as;

COLUMN NAME	ATTRIB	VALUE
FIELD_FLAG	CHAR(1)	N
KEY_FLAG	CHAR(1)	Y
SRCH_FLAG	CHAR(1)	Y
GRP_FLD_FLAG	CHAR(1)	N
MAP_FLAG	CHAR(1)	N
UNMAP_FLAG	CHAR(1)	N
DESC_KEY_FLAG	CHAR(1)	N
ASC_KEY_FLAG	CHAR(1)	Y

Alternative column values could be 0 and 1, or Y and N, or whatever you specify in the FBE. The FBE would be responsible for expanding the BIT settings into appropriate character values during application insert and update calls. It would also

be responsible for collapsing the 8 character values into a single byte field during retrieval processing.

Field Build Exits – data verification

Another possible use of FBEs is for field-level data verification. Suppose you have a field that you want to store in DB2 as a DECIMAL column. DB2 uses MVS Packed Decimal format, with a sign of either “C” for positive numbers or “D” for negative numbers. Assume that the data value for this field is not known when the record is initially added to the file. In some programs, the field is initialised with a valid packed decimal value, but in other programs the field is uninitialised.

VS/2 expects that fields that map to DB2 Decimal columns have valid packed decimal values. If the value of the uninitialised field is not a valid packed decimal number, a data exception error will occur. This will result in an S0C7 abend.

One way to fix this is to identify all the application programs that can update the file, and make sure that they initialise all fields correctly. Alternatively, you can write a VS/2 FBE to verify that the field value is valid. If the value is invalid, the FBE should change it.

Alternatively, the FBE could be used to set uninitialised field values to DB2 null values.

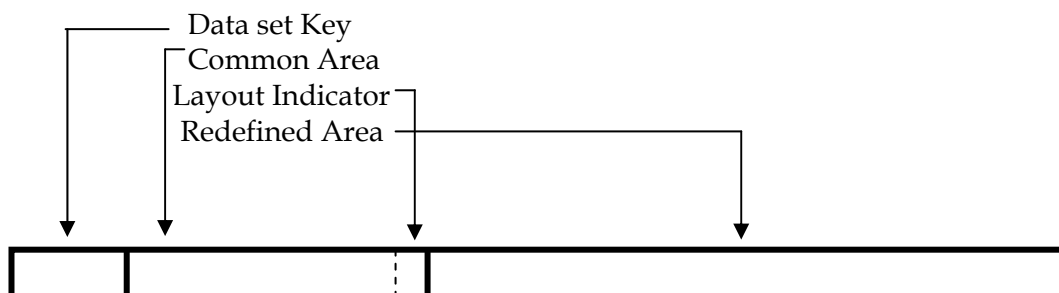
Redefined file structures

In the lifetime of your VSAM files, changes have been necessary to meet new business needs. These range from simple one-line program changes, to more complex changes such as creating additional record structures. The outcome is that a single VSAM file has multiple copybooks for multiple record types.

There is no specific function in VS/2 to handle redefined file structures. However, through user-written exits, it is possible to separate the records into multiple DB2 tables, based on the ability to determine the appropriate record type from some part of the data. (such as the key or a data field).

Because this is a very common situation, it is worth exploring the different options you have in VS/2.

Consider the following VSAM record layout:



The Common Area is common to all records and COPYBOOK sections. It includes a field that indicates the layout type.

There are three different approaches to handling this type of data set

1. The redefined area is stored in a single DB2 column or multiple columns if the record length is more than 255 bytes.
2. There is a single DB2 table that contains columns for all of the possible record layouts. Every DB2 record will include a number of unused columns, depending on the record type.
3. There is one DB2 table containing the columns for the common area, plus one DB2 table per record type. All of these DB2 tables will have the same primary key.

To illustrate these three solutions, let us assume that there are 3 different redefined areas, and that the layout indicator field value can be A, B or C.

The primary table in VS/2

Every VSAM data set must map to a *Primary* DB2 table. VS/2 will use the DB2 primary key of this *Primary* table to navigate the database. The primary key of this table is the key field of the file (for a KSDS) or the relative record number (for an RRDS).

The characteristics of the three potential solutions are discussed next.

Option 1 - Single Table with no column-level granularity

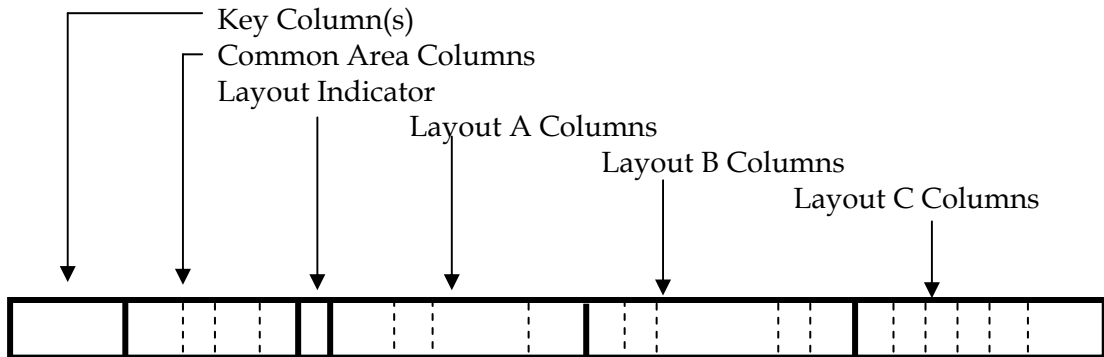
This is the simplest of the 3 solutions, but sacrifices many of the potential benefits of using DB2. The DB2 table looks like this:

```
CREATE TABLE REDEF_OPTION_1
  ( REDEF_FILE_KEY      CHAR ( XX )      NOT NULL
  , COMMON_AREA_COL1_01 CHAR ( XX )      NOT NULL
  , COMMON_AREA_COL1_02 CHAR ( XX )      NOT NULL
  , COMMON_AREA_COL1_03 CHAR ( XX )      NOT NULL
  , LAYOUT_IND          CHAR ( 1 )        NOT NULL
  , BIG_VARIABLE_AREA   CHAR ( XXX )     NOT NULL
  , PRIMARY KEY ( REDEF_SEG_KEY ) ) ;
```

A single column is used to store the redefined area, although this means that the data in this column is of limited value for SQL queries. On the plus side, the performance overhead of this solution is minimal because there will be few columns in the DB2 table, and a single DB2 I/O satisfies the application program data set call.

Option 2 - single table with column-level granularity

This option provides column level granularity in the DB2 table. The layout of the DB2 row will look like the following:



This solution requires a VS/2 FBE exit to handle both retrieval and update calls. The BE tests the layout indicator field to determine the layout of the record being processed and the type of call as follows:

1. For a retrieval call, the exit must determine the record layout type then build the VSAM view of the record using the appropriate DB2 column values.
2. For an update call, the FBE must move the data from the IO area supplied by the application program to the appropriate columns in the DB2 area.
3. No action is required for a DLET call.

A potential drawback of this design may be the limitation of the maximum number of columns allowed in a single DB2 table.

Every row in this table will contain a number of columns without meaningful data values. If you plan to access the table with SQL, either in application programs or with ad hoc queries, you could consider the use of DB2 VIEWS. For example, look at the following DDL.

```
CREATE VIEW LAYOUT_A_VIEW AS
SELECT (common_area_columns,
        layout_indicator column,
        layout_a_columns)
FROM COMPOSITE_TABLE
WHERE layout_indicator_column = 'A');
```

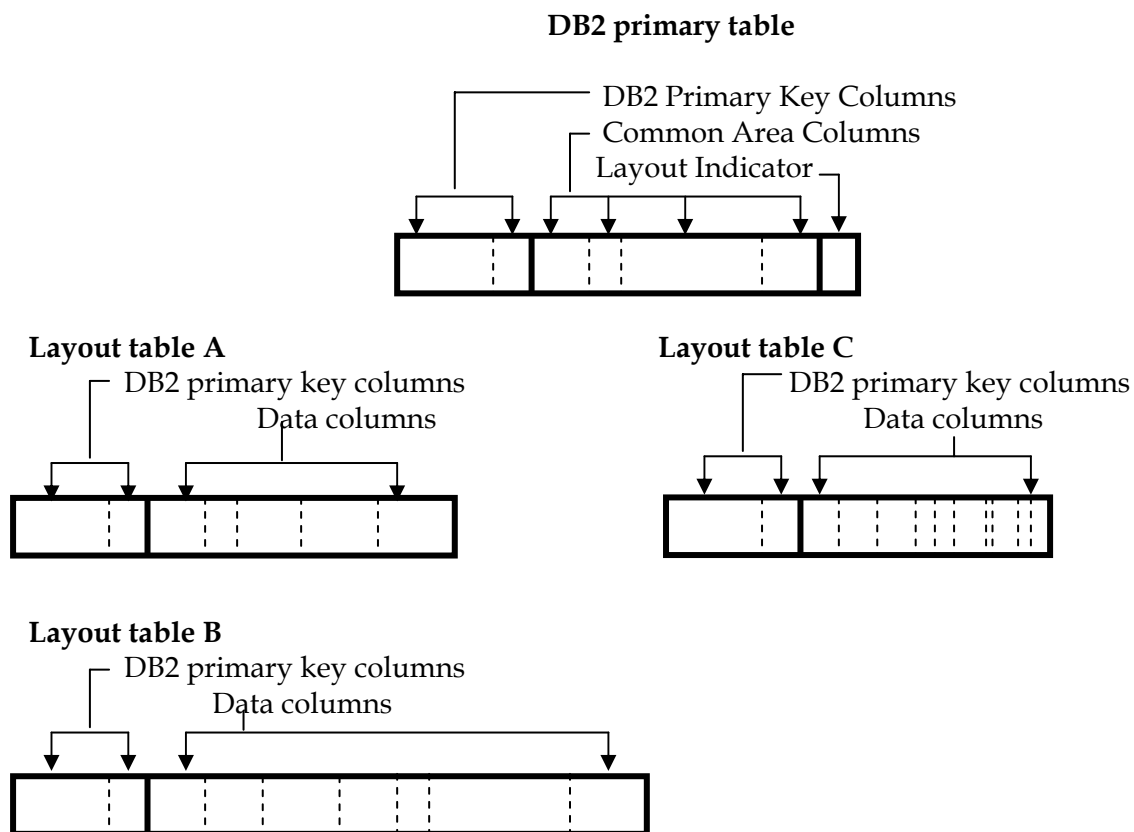
By creating a DB2 VIEW for each layout, the data will be perceived to be in separate tables.

Option 3 - additional tables for each variable copybook layout

This technique is the most commonly used, and requires two VS/2 user exits.

- (a) An FBE is required to build the I/O area for retrieval calls. VS/2 will construct the *common* part of the I/O area from the *primary* DB2 table. Depending on the value of the Layout Indicator, the FBE will issue SQL SELECT for a row in one of the secondary tables, and will then construct the remainder of the record to be returned to the calling program.
- (b) An IRD exit will be required to handle update calls. Depending on the value of the Layout Indicator, the exit will issue SQL INSERT or UPDATE to the appropriate DB2 table. Delete calls will rely in DB2 referential integrity to cascade an SQL DELETE call issued by the VS/2 against the *primary* DB2 table to the appropriate individual layout table. DB2 RI will also ensure the relationship between the table containing the common area columns and the tables for each layout type is maintained.

This is illustrated using the example below.



For each individual layout table, the DB2 primary key columns must match those of the primary table. In addition, a DB2 FOREIGN KEY is defined to establish the relationship from each of the three tables back to the primary table.

As with Option 2, this table design can be improved with DB2 VIEWS. In this case, a predicated JOIN can present a single table image back to the user or application, as shown in the example below.

```
CREATE VIEW LAYOUT_A_VIEW AS
SELECT (P.common_area_columns,
       P.layout_indicator column,
       A.layout_a_columns)
FROM   P.primary_table,
       A.layout_a_table
WHERE  P.primary_key_columns      = A.foreign_key_columns
       AND A.layout_indicator_column = 'A');
```

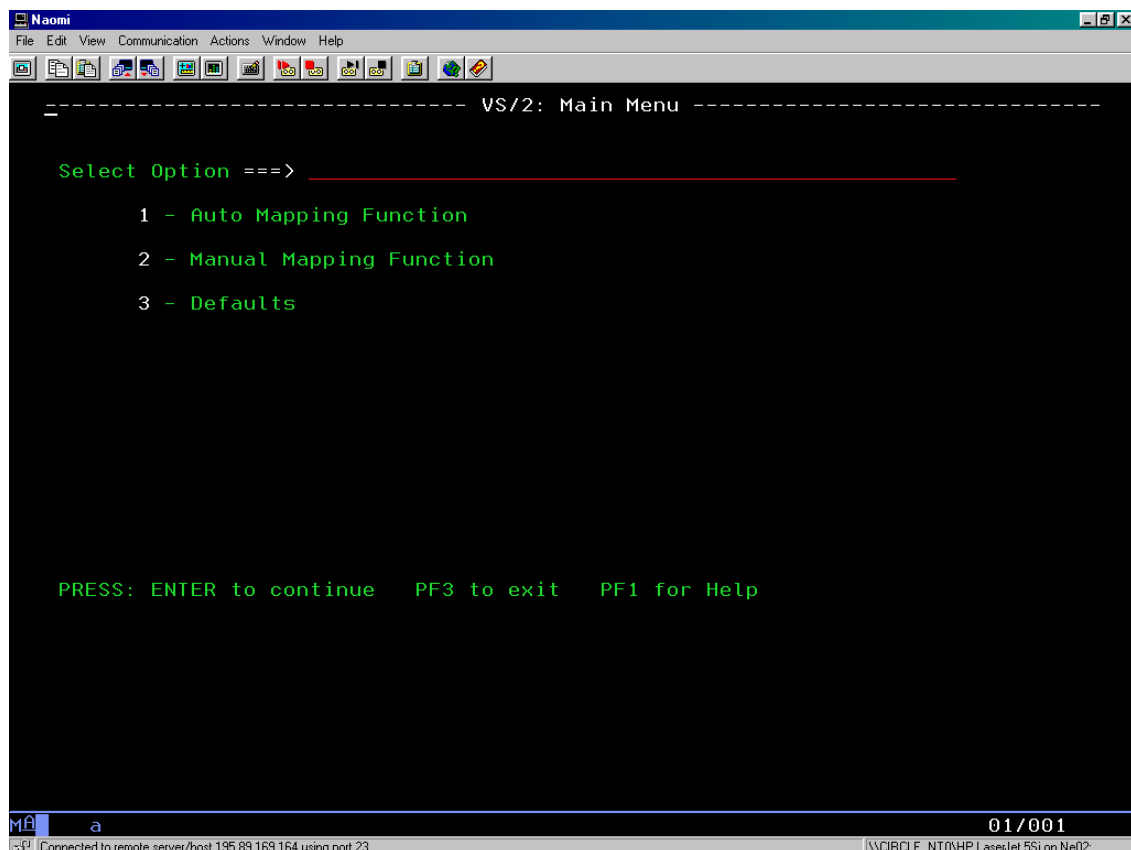
The DB2 views can be used to access the DB2 tables from either new application programs or ad hoc queries. VS/2, in conjunction with the FBE and IRD exits, will use the underlying base tables.

VS/2 main menu

It is difficult to show a tool whose objective is that things remain the same. In other words, VS/2 ensures that your application programs behave and perform the way they have always done, although the data has been migrated.

There follows a selection of ISPF panels from the **VS/2 Mapping Component**. The purpose of this is to help you understand how VS/2 establishes the relationship between the file structure represented in the Copybook, and the DB2 table containing the migrated data.

Firstly, the following is the VS/2 main menu:



Before you map your first file, VS/2 will direct you to Option 3 - Defaults. Here you specify parameter values such as the name of the data set for your VS/2 driver modules, the DB2 ssid and so on. You will only have to specify these defaults once.

You will use the other 2 options throughout the migration process as described next:

Option 1 is used when you are migrating a file to a single VSAM data set. You supply the name of the file copybook and the DDL and mapping data is automatically generated. VS/2 will alert you to copybook situations that require user action. For example, if a copybook field name exceeds 18 characters, VS/2

truncates the DB2 column name and highlights this. A second example is where one or more fields redefine a field. You often see this for date fields, where a single field is redefined as 3 or 4 2-byte fields for CC, YY, MM and DD. Here is an example of the copybook analysis screen:

```

----- VS/2: Edit Columns in STANDIM ----- Row 1 of 10
Command ==> _ Scroll ==> PAGE

PRESS: ENTER to continue  PF3 to exit  PF1 for help

Actions:  R - Rename column          D - Delete column
          U - Update column
          MAP - Enter command to execute automated mapping

Act Position  Column Name      Data Type/Len      Status
-----
- 1          ITEM_NUMBER     CHAR 6
- 7          ITEM_NAME       CHAR 12
- 19         ITEM_COLOUR    CHAR 6
- 25         ITEM_HEIGHT    CHAR 4
- 29         ITEM_COST      DECI 7,2
- 33         ITEM_REORDER_QUANT CHAR 3      Name shortened
- 36         ITEM_SUPPLIER_CODE CHAR 3
- 39         ITEM_DATE_FIRST_SH DECI 9,0    Name shortened
- 44         ITEM_SHELF_LIFE  CHAR 2
- 46         ITEM_DESCRIPTION CHAR 35

***** Bottom of data *****

```

At the bottom of the window, there is a status bar showing 'Mâ a' on the left, '02/015' in the center, and 'Connected to remote server/host:195.89.169.164 using port 23' on the right.

The DB2 column names and types are derived from the copybook. The RENAME option lets you change column name, and the UPDATE option lets you change the data type. The DELETE option is for redefined fields, and to remove unwanted columns (FILLER for example).

Once you have finished making changes, specify MAP. This builds and submits a batch job stream, which creates the DB2 objects and generates run-time driver modules. Alternate indexes are handled implicitly by the automated mapping feature. You can use the SAVE and RESUME commands to save your changes and return to the migration later.

Option 2 is for manually mapping a data set, or for making changes to a data set that was mapped previously either manually or automatically. Data sets that will be migrated to multiple DB2 tables must be mapped manually.

The following is an example of a manual-mapping screen:

```

----- VS/2: List of Fields for VIDKSDS ----- Row 1 of 10
Command ==> _ Scroll ==> PAGE

VSAM Filetype : KSDS      Primary Table   : CIRDL.VID_ITEM
Dataset Length: 00080

Actions: U Update, I Insert, D Delete

Sel Field   Fctn   Build Order Bytes Start  Type Tbl  Column Name  VS/2 Picture
-----
- VIDKEY    KEY    00001 00006 00001  C   P   ITEM_NUMBER
- VIDF001   00002 00012 00007  C   P   ITEM_NAME
- VIDF002   00003 00006 00019  C   P   ITEM_COLOUR
- VIDF003   00004 00004 00025  C   P   ITEM_WEIGHT
- VIDF004   00005 00004 00029  P   P   ITEM_COST
- VIDF005   00006 00003 00033  C   P   ITEM_REORDER_NO
- VIDF006   00007 00003 00036  C   P   ITEM_SUPP_CODE
- VIDF007   00008 00005 00039  P   P   ITEM_DATE_FSHIP  CCYYMMDD
- VIDF008   00009 00002 00044  C   P   ITEM_SHELF_LIFE
- VIDF009   00010 00035 00046  C   P   ITEM_DESCRIPTION
***** Bottom of data *****

```

The file is a KSDS called VIDKSDS and has a record length of 80 bytes. The panel shows that this file maps to the DB2 table ITEM_DETAILS_TAB. Down the left hand side of the bottom half of the screen you see the column heading **Field**, and below that the fields called VIDKEY, VIDF0001, VIDF0002 and so on. The actual field names are irrelevant; what is important is the length of the field, shown in the column **Bytes**, and its position in the record, shown in the column **Start**.

Down the right hand side of the bottom half of the screen is the **DB2 Column Name**. When VS/2 processes a retrieval call from an application program, the data in the DB2 column ITEM_NUMBER will be placed at offset 1 in the record area; the ITEM_NAME column data will be placed at offset 7, and so on for the entire 80 bytes.

VS/2 uses this mapping information to generate a DIM driver module. There is one DIM for each migrated VSAM file. At run-time, VS/2 will load the appropriate DIM(s), so therefore there is no run-time access to the VS/2 mapping tables.

A typical VS/2 migration project team

In a typical installation the mapping is controlled at a single point, often in the DBA group. They liaise with the application groups who are responsible for the application file(s) being migrated.

They also interface to the CICS systems programmers, who are responsible for defining the VS/2 objects in CICS. They also interface to the Operations group, or whoever is responsible for maintaining the JCL for batch jobs.

A typical VS/2 migration project is made up of the following full-time members;

- The DBA group, who are responsible for creating the DB2 objects, the VS/2 mapping and the data migration.
- Application programmers, who are responsible for the initial analysis, defining data re-engineering requirements, coding VS/2 exits and application testing

Part-time support to the migration is required from the CICS support group and Operations support.

Production cutover is largely a data migration issue, and is normally handled by the DBA group. Post production performance and tuning is also a DBA responsibility.

Preparing your applications for VS/2

Part of the process of migrating a VSAM data set is to enable access to it from your application programs. The way this is done differs between batch and CICS, both of which are explained next.

Accessing migrated data sets from batch programs

To access a migrated data set from a batch program requires some simple JCL changes. These are;

- (a) The VS/2 distribution library and the library containing the DIM and DDM drivers must be added to the STEPLIB or JOBLIB.
- (b) The DD statement for the migrated VSAM file must be replaced with the following;

```
//vsamfile DD SUBSYS=(ssi,db2id,DIM name)
```

Where	ssi	=	the name of the VS/2 batch subsystem set at VS/2 installation time
	db2id	=	The DB2 subsystem name
	DIM name	=	The name of the VS/2 DIM for the migrated file

- (c) Optionally, a new DD statement to enable you to specify an alternative DB2 plan name (the default plan name is the DIM-name)

VS/2 uses these parameters to establish a CAF connection to DB2. When an application program accesses more than one migrated data set, VS.2 can either use a single DB2 thread or multiple threads.

Accessing migrated data sets from CICS programs

There are three aspects to be addressed to allow CICS programs to access VS/2 migrated VSAM data sets.

- (a) The VS/2 distribution library and the library containing the DIM and DDM drivers must be added to the DFHRPL DD statement.
- (b) The VS/2 DIM and DDM drivers for each migrated data set must be defined in the CICS PPT
- (c) A VS/2 data set control table (DST) handles access to migrated data sets from CICS programs. Macros are supplied to assemble the table, and a CICS transaction is provided to allow you to introduce an updated table.

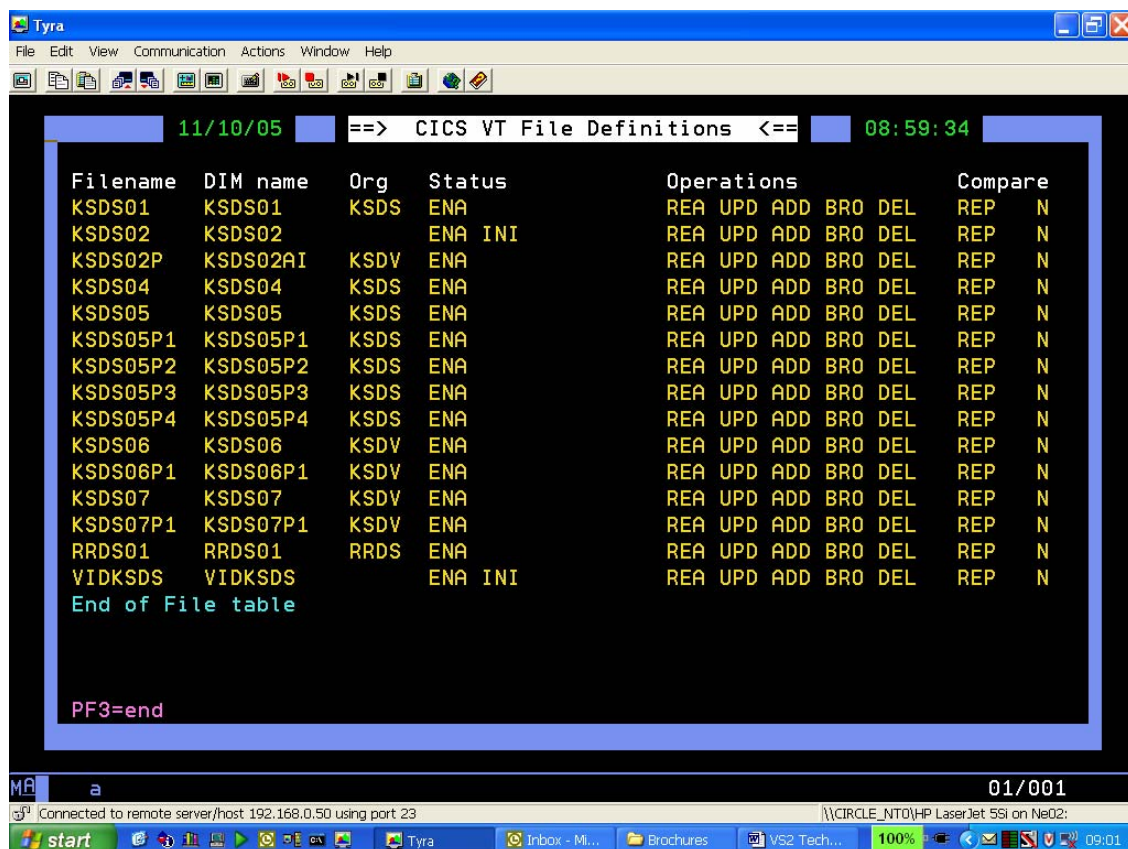
The VS/2 DST table input looks like this;

```

VID#FTAB VS2TAB      TYPE=INITIAL
          VS2TAB      TYPE=ENTRY ,
                               FILE=ITEMDET ,
                               DIM=ITEMDIM ,
                               STATUS=ENA ,
                               OPEN=S ,
                               ADD=YES ,
                               BRO=YES ,
                               DEL=YES ,
                               REA=YES ,
                               UPD=YES ,
                               RESET=YES ,
                               RECORDF=FIX
          VS2TAB      TYPE=ENTRY ,
                               FILE=INVOICE ,
                               DIM=INVOICE ,
                               STATUS=ENA ,
          .
          .
          .
          VS2TAB      RECORDF=VAR
          VS2TAB      TYPE=FINAL
          END

```

You can see the status of all migrated data sets using the VS2D CICS transaction. A sample screen follows:



The filename in this screen is the name of the VSAM file as defined to CICS. The files KSDS001 and KSDS03 have a status of **UNE**, which means that they are unavailable to

CICS application programs. This status is automatically controlled by the standard CICS CEMT transaction.

VS/2 debugging and trace facilities

A number of trace and debugging facilities are provided with VS/2. Their prime usage is during data conversion and initial testing.

Interpreting bad SQL codes

VS/2 is unable to translate certain DB2 error conditions into appropriate and meaningful VSAM return codes and reason codes. For example, if VS/2 tries to process a DB2 table and encounters a -904 SQL code (unavailable resource), VS/2 will return an error to indicate that the VSAM call was not successful.

The way this is handled by VS/2 is as follows;

- ❑ For batch programs, the closest fit VSAM return code and reason code is returned to the application program. At the same time, the SQL code is formatted by VS/2 and returned in the job message log.
- ❑ In CICS, the closest fit CICS condition is set. The SQL error code is formatted and returned to the CICS DD statement VS/2DMP.

In most installations, SQL error conditions such as -904 are rare in a Production environment. However, it is recommended that the VS/2DMP DD statement is added to every CICS system that uses VS/2.

VS/2 trace facility

VS/2 includes a trace facility that provides a range of information, from a high-level overview to a complete detailed trace of the VS/2 activity that was executed to service VSAM calls. This information is especially valuable during initial data conversion to help resolve potential mapping inconsistencies or errors, and to help you verify that your user exits are functioning correctly. The trace facility captures and formats the contents of various VS/2 control blocks at the following trace points within the VS/2 interface module:

- | | |
|----------|---|
| Point 1: | The application call |
| Point 2: | Internal Diagnostic data |
| Point 3: | DB2 OPEN CURSOR |
| Point 4: | DB2 FETCH |
| Point 5: | DB2 CLOSE CURSOR |
| Point 6: | The value for each field built from the row retrieved from DB2 |
| Point 7: | The value for each DB2 column built from the record to be updated |
| Point 8: | The VS/2 internal return code |
| Point 9: | After a DB2 INSERT, UPDATE or DELETE call has been processed. |

You can control the information that is included in the trace by selecting the appropriate trace points.

Activating the trace for batch

The following are the additional DD statements you need to add to your JCL to provide VS/2 tracing for a batch program, plus sample control statements.

```
//VIDTRCE DD SYSOUT=*,RECFM=FBA,LRECL=133
//VIDTRCEP DD *
TRACE P01,P02,P03,P04,P05,P06,P08,P09,P07,DIM=ITEMDET
TRACSET CALLS=000035,SKIP=000360
```

These control statements define that the tracing is to be performed for the file ITEMDET. The first 360 calls in the program will not be traced, calls 361 to 395 will be traced.

Tracing can be activated for up to 10 files at a time.

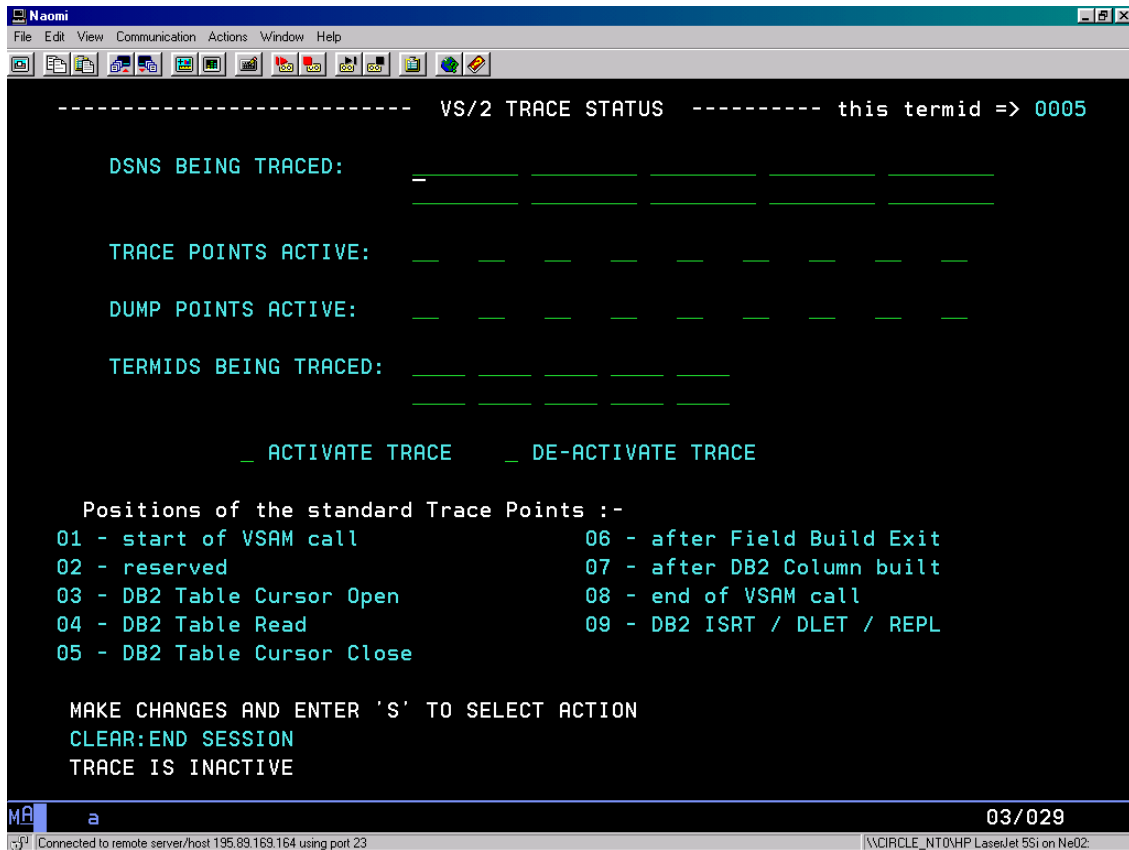
A sample VS/2 trace is shown next.

```
Naomi
File Edit View Communication Actions Window Help
-----
Display Filter View Print Options Help
-----
SDSF OUTPUT DISPLAY CIRDLSHB JOB08921 DSID 104 LINE 78 COLUMNS 02- 81
COMMAND INPUT ==> SCROLL ==> CSR
***TRACE POINT 1 - START OF VSAM CALL REQ = 0000004 TIME =
CALL TYPE = PUT , DIM = VIDKSDS , DDM = VIDKSDS# , DSNTYPE = KSDS
DD = KSDS0001 , RPLNUM = 001 , POS = N , DIR = F
KEYUSE = 3 , KEY VALUE = 243001 F2F4F3F0F0F1
***TRACE POINT 9 - DB2 ISRT / DLET / REPL TIME =
TABLE = VID_ITEM SQL RETURN = 000
999999NEW.MUDGET..PUCE.. F9F9F9F9F9F9D5C5E640D4E4
7777.....7772002-04-237 F7F7F7F70000700C004DF7F7
7ALL.THE.SEVENS..... F7C1D3D340E3C8C540E2C5E5
.....243001..... 404040404040404040404040
..... 000000000000
***TRACE POINT 8 - END OF VSAM CALL RC = 0000 TIME =
***TRACE POINT 1 - START OF VSAM CALL REQ = 0000005 TIME =
CALL TYPE = GETU , DIM = VIDKSDS , DDM = VIDKSDS# , DSNTYPE = KSDS
DD = KSDS0001 , RPLNUM = 001 , POS = N , DIR = F
KEYUSE = 1 , KEY VALUE = 999999 F9F9F9F9F9F9
***TRACE POINT 5 - DB2 TABLE CURSOR CLOSE DIMNAME VIDKSDS CURSOR NO=00 TIME =
TABLE = VID_ITEM SQL RETURN = 000
***TRACE POINT 3 - DB2 TABLE CURSOR OPEN DIMNAME VIDKSDS CURSOR NO=01 TIME =
TABLE = VID_ITEM SQL RETURN = 000
KEY HOST VARIABLES = 999999 F9F9F9F9F9F9
Má a 22/074
Connected to remote server/host.195.89.163.164 using port 23 | \VCIRCLE_NT\NHP LaserJet 5Si on Ne02
```

The trace shows a PUT call, and the resulting DB2 INSERT. The first few trace records for a GET call are also shown.

Activating the trace in CICS

CICS tracing is activated by the VS2T transaction, which presents the following screen:



```
----- VS/2 TRACE STATUS ----- this termid => 0005

DSNS BEING TRACED:  _____
                   _____

TRACE POINTS ACTIVE:  _____

DUMP POINTS ACTIVE:  _____

TERMIDS BEING TRACED:  _____
                   _____

      _ ACTIVATE TRACE      _ DE-ACTIVATE TRACE

Positions of the standard Trace Points :-
01 - start of VSAM call           06 - after Field Build Exit
02 - reserved                     07 - after DB2 Column built
03 - DB2 Table Cursor Open       08 - end of VSAM call
04 - DB2 Table Read              09 - DB2 ISRT / DLET / REPL
05 - DB2 Table Cursor Close

MAKE CHANGES AND ENTER 'S' TO SELECT ACTION
CLEAR:END SESSION
TRACE IS INACTIVE
```

MA a 03/029

Connected to remote server/host:195.89.169.164 using port 23 | \CIRCLE_NT\HP LaserJet 5Si on Ne02

As you can see, up to 10 files can be traced simultaneously.

Installing VS/2

You install VS/2 in your system with a few straightforward operations, as follows:

- ❑ Download the VS/2 installation tape
- ❑ Run the customisation application
- ❑ Create the VS/2 control DB2 tables
- ❑ Bind the VS/2 DBRMs
- ❑ Make the VS/2 dialog available from the ISPF menus
- ❑ Define the VS/2 batch MVS subsystem
- ❑ Define the VS/2 resources to CICS jobs

Each of these operations is now described. VS/2 supplies JCL and other information in the sample library to make them as easy as possible.

Download installation tape

VS/2 is supplied either electronically as a .ZIP file, or on a standard labelled tape containing twelve data sets, as follows:

LIBRARY NAME	CONTENTS
VS2.LOAD	Executable load modules
VS2.SAMPLIB	Sample library
VS2.ISPCLIST	ISPF CLIST
VS2.ISPLLIB	ISPF programs
VS2.ISPMLIB	ISPF messages
VS2.ISPPLIB	ISPF panels
VS2.ISPSLIB	ISPF skeletons
VS2.DBRMLIB	DBRMs
VS2.INSTALL	Installation jobs
VS2.JCLLIB	JCL library
VS2.LINKLIB	VS/2 subsystem modules
VS2.LPALIB	VS/2 subsystem modules

File 10 of the VS/2 installation tape is an IEBCOPY version of a PDS that contains JCL to download the other data sets. The JCL is also shown in the *VS/2 User's Guide*. You can extract VS2.JCLLIB by using the IBM utility IEBGENER.

Run the customisation application

An ISPF application will customise all of the necessary INSTALL and SAMPLIB library members. You supply local data set names, DB2 sub-system characteristics and jobcard formats. A README member containing all of your supplied parameter values is created for your reference.

Create DB2 tables for the VS/2 dialogs

VS/2 uses DB2 tables to contain the VSAM to DB2 mapping information that you define with the VS/2 dialogs. You can create these DB2 tables by using the JCL supplied in member CRVS/2TBL on VS2.INSTALL. You may need to change the database and tablespace names to suit your organisation's standards.

Bind VS/2 DBRMs

You use the JCL in member VIDBIND on VS2.INSTALL to bind the DB2 plans used by various VS/2 functions. You will need to modify the JCL to suit your requirements before submitting it. You will also need BINDADD authority on DB2.

You must perform GRANT EXECUTE for these plans to authorise nominated users to the VS/2 dialogs.

Make VS/2 dialogs available through ISPF

To make the VS/2 dialog available to the VS/2 project team, you must copy the REXX EXECs in the VS2.ISPCLIST data set to one of the CLIST or REXX libraries allocated to your TSO session under the SYSPROC DD name.

You should also copy the JCL procedures VIDASM, VIDDDMG, VIDLBR, VIDPRE, VIDDIMG and VIDSTGII the VS2.SAMPLIB data set to a JES procedure library.

Define VS/2 resources to CICS

Earlier, the steps to enable VS/2 access from CICS programs were described. There are a number of installation activities that must be completed first.

- ❑ Add the VS/2 program VID#INIT to the PLT
- ❑ Run the VS2.INSTALL member CICSDEF to define all the VS/2 system objects to CICS (this uses the DFHCSDUP batch CSD update program). Add the VS/2 group to an active group list.
- ❑ Assemble the VS/2 DST table
- ❑ Add the VS/2 distribution load library, plus the application drivers library to the CICS RPL

- ❑ Add the VS2TRC DD statement
- ❑ Optionally add VS/2 trace DD statements to the CICS JCL
- ❑ Run the VS/2 installation checker transaction VS/2R

If CICS does not already access DB2, additional installation activities are required. These will be described in the appropriate DB2 manual.

Install the VS/2 subsystem

VS/2 uses an MVS subsystem to access migrated VSAM files from batch programs. If you are using OS/390 V2.4 or later, you can initially install the VS/2 subsystem dynamically without an IPL.

There are instructions for installing the subsystem dynamically and permanently. An MVS Systems Programmer will be familiar with both methods.

Run the VS/2 IVP

There is a batch and a CICS IVP. Both use a file called KSDS001, which exists in DB2. The DB2 data (16 rows) is inserted by the installation job CRVS2TBL. The batch part consists of one Cobol and one PL/1 program. Each program produces a short report file.

The CICS IVP uses the supplied transaction VS2V, and requires you to supply an input key value that is documented in the manual.

CONTACT US

At Circle Computer Group, we are always looking for ways to improve any aspect of our software, the documentation and the support we provide to our customers. Therefore, we welcome any comments from you regarding any of these aspects, or any other related matter.

There are a number of alternative ways that you can contact us.

By Telephone	+44-01428 664500
By Fax	+44-1428 664509
By e-mail	migration@circle-group.com
By Post	Circle Computer Group Harrow House 23 West Street Haslemere Surrey GU27 2AB UK

You can check out the full range of Circle's products and services by logging on to www.circle-group.com.