Abstract

Worldwide, a vast amount of data is maintained in DL/I data bases. Version 1 Release 1 of the SAS/IMS-DL/I product provided read and update access to this data from a SAS data step through an IMS/VS or CICS/OS/VS batch DL/I region. Version 1 Release 2 of the SAS/IMS-DL/I product expands this access to "online" data bases and IMS/VS message queues through a Batch Message Processing DL/I region. This Release also provides installation control of SAS/IMS-DL/I usage. In the near future an update to Release 2 will provide the facility to access CICS/OS/VS "online" data bases through a CICS Shared DL/I region.

DL/I Description

DL/I is a hierarchic data structure definition and access language. Data are stored in segments at the node points of the tree structure. Various DL/I access methods are layered over OS/VS access methods to store the data in OS/VS data sets. A CALL-level access language provides a PL/I, COBOL, or Assembler (BAL) programmer the ability to retrieve and update segments either sequentially or randomly in the data base.

A Data Base Definition (DBD) defines a physical hierarchic structure. It maps the hierarchic structure to OS/VS data sets through the DL/I access methods. A logical DBD maps a new hierarchic structure from one or more physical DBDs which are logically related.

A Program Specification Block (PSB) defines the logical hierarchic structures as seen by the program in Program Control Blocks (PCB). The logical hierarchic structure may be all or some root-connected subtree of a hierarchic structure defined in a DBD. Each PCB also defines the access authority which the program has to each segment in the logical hierarchic structure.

Most DL/I installations have one or more Data Base Administrators whose function is to control and optimize these structures. Generally, a programmer is only concerned with the logical view of the data in the PSB.

Batch DL/I Region Description

A simplified flow of control and information is depicted in FIGURE 1. OS/VS JCL executes the DL/I region controller and specifies both the PSB and the name of the user program. The region controller establishes the DL/I environment and invokes the user program. The user program may access OS/VS data sets through normal language constructs. The program accesses DL/I data bases through a CALL-level interface to DL/I.

In the CALL parameters, the user program specifies:

1) A function to perform (e.g. 'ISRT' for insert)
2) A PCB in the PSB
3) An I/O Area
4) Optional qualification data in up to 15 Segment Search Arguments (SSA)s.

FIGURE 1
SAS/IMS-DL/I Version 1 Release 1

Version 1 Release 1 of the SAS/IMS-DL/I product provides extensions to the SAS DATA step language to allow DATA steps to execute DL/I CALLS in order to access DL/I data bases using batch DL/I region facilities.

FIGURE 2 depicts the flow of control and information in a SAS DATA step which accesses DL/I data bases. SAS is executed in either batch mode with standard OS/VS JCL or interactively under TSO. A SAS DATA step is compiled and executed under SAS. If the DATA step uses SAS/IMS-DL/I language extensions to access DL/I data bases, SAS attaches a batch DL/I region specifying a PSB and a SAS/IMS-DL/I interface module as the user program. The name of the PSB is specified by the user via DATA step language extensions. The DATA step program issues DL/I calls through an INPUT/PUT interface to the SAS/IMS-DL/I interface module. The interface module converts the INPUT/PUT parameters to DL/I call parameters and CALLS DL/I. Sample programs at the end of this paper illustrate the usage of these statements. Non-DL/I INFILE, INPUT, FILE, and PUT statements may be interspersed in the DATA step to access OS/VS data sets. SET, UPDATE, MERGE, and OUTPUT statements may be used to access SAS data sets.

The INFILE statement extensions specify the PSB used to communicate with DL/I in the DATA step, may specify values and variables used to build the DL/I call, and may specify variables to receive DL/I call feedback. The INPUT statement is used to execute a DL/I "get" type call. The FILE statement associates the PSB, values, and variables in the DL/I INFILE with subsequent PUT statements. The PUT statement is used to execute a DL/I "update" type call.

FIGURE 2
IMS/VS Online System Description

The component regions of an online IMS/VS system are depicted in FIGURE 3. In an online IMS/VS system there is one and only one control region. There may be multiple Message Processing Program (MPP) Regions and multiple Batch Message Processing (BMP) Regions.

The Control Region executes under OS/VS to control access to system terminals and data bases. The data communication functions control the terminals with a message queuing system. The data base functions control data base access with a record level enqueue/dequeue mechanism. The supervisor functions log all systems events to the IMS LOG for recovery/restart purposes and schedules application programs in the dependent regions to process terminal transactions. No application program executes in the Control Region.

An MPP Region executes under OS/VS and establishes communication with the Control Region. The Control Region schedules application programs in an MPP Region to process transactions in the message queues. The application program accesses the system message queues and data bases through the Control Region. The program cannot access OS/VS data sets unless the appropriate DD statements were included in the MPP region JCL (this is not frequently done).

A BMP Region also executes under OS/VS and establishes communication with the control region. The PSB and program to be executed are specified on the JCL EXEC statement. The JCL typically includes DD statements for OS/VS data sets to be accessed by the application program. The program accesses the IMS/VS system message queues and data bases through the Control Region.

In an online IMS/VS system more than one program may access data bases and message queues simultaneously. The access is controlled by the Control Region to insure data integrity and recoverability. Until the latest release of IMS, data bases could not be shared by the Control Region and batch regions with integrity. The latest IMS Release does allow this simultaneous access with integrity.

![IMS/VS Online System Diagram](image-url)
SAS/IMS-DL/I Version 1 Release 2

The two major functional enhancements of SAS/IMS-DL/I Version 1 Release 2 are:

1) The ability to access the message queues and data bases of an online IMS/VS system through a BMP Region,

2) The ability to control and tailor the Capabilities of SAS/IMS-DL/I at each installation.

The SAS and IMS system environment, control flow, and data flow are depicted in FIGURE 4. When executing a SAS DATA step which accesses online IMS system resources through a BMP Region the control flow within SAS is identical to the batch DL/I version except that a BMP Region is attached. The SAS/IMS-DL/I interface module has been modified for Release 2 to allow message queue access calls. Some system service calls such as BASIC CHK are also supported.

Release 2 provides the ability for each installation to control and tailor the use of SAS/IMS-DL/I with SAS parameters.

DL/I Region execution parameters are specified with SAS parameters. Additional SAS parameters for online and batch region types may limit the DL/I access calls allowed to read only types. These SAS parameters may be assigned installation defaults so that the programmer does not have to worry with most of them. In addition, the individual SAS parameters may be restricted such that the installation default may not be altered in a program.

In the near future, an update to Release 2 will allow access to "online" CICS/OS/VS DL/I data bases through a CICS Shared DL/I region. General purpose SAS/IMS-DL/I utility programs are planned for inclusion in the SAS sample library.

Version 2 of SAS/IMS-DL/I is presently planned to contain enhancements in two areas. One area will address the IMS technician who would like a more PL/I-like SAS/IMS-DL/I DATA step language. The other area will address the non-IMS programmer who wants to obtain data from complex DL/I data base structures without learning the intricacies of IMS programming.

FIGURE 4

SAS/IMS-DL/I BMP EXECUTION
Sample Programs

The three sample programs below access one or both of the data base structures pictured in FIGURE 5.

**SAS/IMS-DL/I EXAMPLE 1**

**PURPOSE:**
Create a SAS Data Set containing one observation for each root segment, SEG11, of the Data Base, DB1. Each observation will contain SAS variables whose values are obtained from data in SEG11.

**ASSUMPTION:**
PSB1 is sensitive to SEG11 only in the first PCB of the PSB.

**PROGRAM:**

```sas
DATA SEG11DAT;
INFILE PSB1 DL1;
INPUT @x1 var1 fmt1 ..@xn varn fmtn;
```

**DESCRIPTION:**
PSB1 is identified as the PSB used to access data bases in the INFILE statement. When no other INFILE options are specified, SAS/IMS-DL/I defaults all call functions to 'GN' and the PCB specification to the first PCB in the PSB. Therefore, each execution of the INPUT statement issues a 'GN' call against the first PCB in the PSB with no segment qualification. Since the PSB is only sensitive to SEG11 in the first PCB, each SEG11 segment will be returned sequentially. When the end of the data base is encountered, DL/I returns a 'GB' status code to SAS/IMS-DL/I which indicates the END-OF-FILE condition since only sequential calls have been made.

**SAS/IMS-DL/I EXAMPLE 2**

**PURPOSE:**
Create a SAS Data Set containing one observation for each dependent segment, SEG13, in the data base, DB1. Each observation will contain SAS variables whose values are obtained from data in the root segment, SEG11, or SEG13.

**ASSUMPTION:**
PSB2 is sensitive to all the segments of DB1 and allows a path call at the root level.
PROGRAM:

DATA SEG1113;
LENGTH SSA1 $11 SSA2 $9 DB $8;
INFILE PSB2 DLI DBNAME=DB
SSA=(SSA1, SSA2);
DB=DB1;
SSA1=SEG11 *D;
SSA2=SEG13;
INPUT @I var1 fmt1 ... @x var fmtn;

DESCRIPTION:

This program accesses the data base sequentially in much the same manner as Example 1. However, the program does not depend on the first PCB in the PSB being sensitive to the desired data base and segments. Therefore, the program uses the DBNAME and SSA options on the INFILE statement.

The DBNAME=variable specified, DB, is assigned the name of the data base to be accessed prior to the INPUT statement. The name is used to select the proper PCB in the PSB.

The SSA=variables specified SSA1 and SSA2, are assigned DL/I segment search argument values to qualify the 'GN' call which is issued with the INPUT statement. SSA1 specifies that the SEG11 data segment is returned in the input buffer by use of the *D command code. SSA2 specifies that the SEG13 data segment is also returned in the input buffer following the SEG11 data.

The INPUT statement must be coded to obtain the desired data values for this observation knowing that the input buffer contains SEG11 data followed by SEG13 data.

SAS/IMS-DL/I EXAMPLE 3

PURPOSE:

Update SEG22 with data given in SEG13.

ASSUMPTIONS:

PSB3 is sensitive to all segments of DB1 and DB2 and update sensitive to SEG22.

Each SEG13 contains a value for KEY21 and KEY22, a value to replace some data in SEG22, and a value to add to a data field in SEG22.

PROGRAM:

DATA NULL;
LENGTH SSA1 SSA2 $30 DB $8;
INFILE PSB3 DLI CALL=FUNC
STATUS=ST DBNAME=DB SSA=(SSA1, SSA2);
FUNC='GN'; SSA1=SEG13;
DB=DB1; SSA2='';
INPUT @;
IF _ERROR_ THEN DO;
IF 'ST'='GB' THEN DO;
ERROR=0; STOP; END;
ELSE ABORT ABEND;
INPUT @10 KEY21 $4. @14 KEY22 $6.;
@20 CHNG $25. @45 INCR IB4.;
DB=DB2; FUNC='GHU';
SSA1=SEG21 (KEY21 = '1IKEY2111');
SSA2=SEG22 (KEY22 = '1IKEY2211');
INPUT @;
IF _ERROR_ THEN DO;
IF 'ST'='GE' THEN DO;
ERROR =0; FILE LOG;
PUT 'NO DBR WITH' KEY21= KEY22=;
RETURN; END;
ELSE ABORT ABEND; END;
INPUT @55 OLDAT IB4.;
FILE PSB3 DLI;
SSA1=''; SSA2=''; FUNC='REPL';
PUT _INFILE_ @;
NEWDAT=OLDAT+INCR;
PUT @55 NEWDAT IB4.;
@15 CHNG;
IF _ERROR_ THEN ABORT ABEND;

In this program, we randomly update SEG22 data with update data obtained by sequentially reading SEG13. Since the program is issuing more than simple 'GN' calls, we specify a variable, FUNC, with the CALL= option to contain the DL/I function when an INPUT or PUT statement against the DL/I PSB is executed. The STATUS= variable specified, ST, contains the DL/I status code returned from the last DL/I call.

The first INPUT statement in the DATA step executes a 'GN' call for the SEG13 data. The trailing @ holds the data in the buffer for the next INPUT statement. Before data is transferred to the SAS variables with the next INPUT statement, the program checks to see if a potential error has occurred. Since other DL/I calls in the program specify random access by qualifying an SSA with search or key value, SAS/IMS-DL/I does not flag the END-OF-FILE condition when DL/I returns a 'GB' status code. The program checks for this 'error' condition. When the condition arises, the program resets the _ERROR_ flag and stops the DATA step execution since all SEG13s have been
processed. If any other error condition has been flagged, the program aborts with an abend. Note that it would be wise to test for errors in the prior examples but it is not required to stop the DATA step.

The second INPUT statement obtains the update data from the INPUT buffer. A fully qualified 'GHU' call is formatted to retrieve the SEG22 segment with the next INPUT @ statement. Again, the ERROR and ST variables are checked to handle special situations. In this case, a 'GE' status code indicates that the SEG22 statement does not exist.

When no error conditions arise, the data field to be incremented is obtained from the INPUT buffer. The replace call is formatted and the PSB is flagged as the current PUT destination with the FILE PSB3 DL/I; statement. The PUT buffer is preformatted with the contents of the INPUT buffer by the PUT INFILE @; statement. The new value for the incremented field is inserted in the PUT buffer along with the data which replaces existing data by the final PUT statement. This statement also executes the DL/I call since there is no trailing @. Any error from the update call aborts the DATA step with an abend.
INFORMATION SYSTEMS
General Issues in Information Systems