

Advantage™ VISION:Transact® for CICS®

Environment Guide

Release 7.8



Computer Associates®

B02572-1E

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

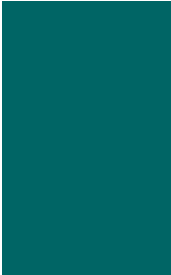
The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227- 19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2004 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.



Contents

Chapter 1: Introduction

Overview	1-1
Documentation	1-1
How This Guide Is Organized	1-2

Chapter 2: System Overview

Description	2-1
Development	2-1
Data Definitions, Screen or COMMAREA Processing Definitions, and Cataloged Requests	2-2
Application Generation	2-2
Application Installation	2-3
Executing Your Application	2-4

Chapter 3: Definition and Source Statement Retrieval Runs

Definition Run	3-1
Source Statement Retrieval Run	3-4
Multiple M5LIBs	3-6

Chapter 4: Application Generation Run

Overview	4-1
Application Generation Flow and JCL	4-2
Multiple M5LIBs	4-12

Chapter 5: Application Installation

Overview	5-1
Processing Screen Formatting Source Statements	5-2
IMS TM MFS Utility JCL	5-2
CICS TS BMS Map Generation	5-3
Application Link Edit	5-4
Defining the Generated Application to the Target Communications Monitor	5-5
IMS TM APPLCTN and TRANSACT Macro Generation	5-7
CICS TS CSD Updates	5-8
PSB Generation	5-11
ACB Generation	5-12
Binding Your Application For DB2	5-12

Chapter 6: Application Execution

Overview	6-1
Application Execution Under IMS TM	6-1
Additional IMS TM Commands	6-3
VISION:Transact Generated Applications and Existing IMS TM Applications and Subroutines	6-3
Execution Tracing of IMS TM Applications	6-4
Application Execution Under CICS TS	6-5
CICS TS Commands	6-7
VISION:Transact Generated Applications and Existing CICS TS Applications and Subroutines	6-7
Execution Tracing of CICS TS Applications	6-7
Messages Issued During Application Execution	6-10
Terminal Support	6-11
COMMAREA Support	6-11

Chapter 7: Characteristics of the Generated Application

Application Structure	7-1
Processing Common to Generated Applications	7-3
Initial Entry to a Transaction	7-3
Redisplaying the Input Screen	7-5
Circumstances Under Which a Record or Segment is Re-read	7-5
IMS TM PCB Selection	7-6
Standard Data Base PCB Selection	7-6
Positional Data Base PCB Selection	7-7
IMS DB Processing	7-8
Relational Data Base (DB2) Processing	7-9

IMS TM PROCESSING	7-9
CALL, DLI, and PLI Statements (IMS TM)	7-12
Calling Assembly Language Subroutines (IMS TM)	7-13
Calling PL/I Subroutines (IMS TM)	7-14
Calling COBOL Subroutines (IMS TM)	7-14
Passing Control to Another IMS TM Transaction	7-15
MFS Features Supported	7-16
Message Format and Attributes	7-17
CICS TS PROCESSING	7-18
CALL and DLI Statements (CICS TS)	7-21
Passing Control to Another CICS TS Transaction	7-22
BMS Features Supported	7-24
Processing Clear and PA Keys	7-25
Considerations For The Development Of Applications For Multiple Targets	7-26

Chapter 8: Library Utilities

Overview	8-1
The M5UTIL Program	8-1
M5UTIL Initialization	8-3
M5UTIL Dump	8-4
M5UTIL Restore	8-5
M5UTIL Condense	8-6
M5UTIL Copy	8-7
M5UTIL Merge	8-9
M5UTIL and Multiple UC Statements	8-11
Other Utility Programs	8-12
Defining a VSAM M5LIB	8-12
Library Initialization Utility	8-13
Library Condense Utility	8-14
Library Dump and Restore Utilities	8-16
Libraries On Shared DASD	8-17

Appendix A: Reading VISION:Transact Application Dumps

Introduction	A-1
Overview of Data Areas	A-1
M5COM	A-1
GOODSTUF	A-1
DBDATBL	A-2
SEGTABLE	A-2
PREFABL	A-3
SKEYHDR	A-5
SKEYBL	A-5
DBWORKBL	A-5
TPDATBL	A-5
USERFLAG	A-5
Techniques	A-6
Locating M5COM	A-6
Locating Working Storage	A-6
Locating Flag Data	A-6
Locating Temporary Fields	A-6
Locating Data Base Segments	A-7

Index

Introduction

Overview

This Environment Guide describes how to use the VISION:Transact generator to build online applications. This guide also provides information about the installation and execution of applications generated with VISION:Transact.

VISION:Transact is an implementation system designed to enable you to generate online applications to execute under IMS TM or CICS TS.

Documentation

In addition to this guide, the following documents are available:

- The *Advantage VISION:Transact Installation Guide* explains how to install and maintain the VISION:Transact system and tailor it to your unique requirements.
- The *Advantage VISION:Transact Specifications Guide* describes source specifications for all VISION:Transact generated applications.
- The *Advantage VISION:Transact Reference Guide* explains how VISION:Transact generated applications work. It provides a conceptual approach to application development using VISION:Transact and the flow of generated applications. The *Advantage VISION:Transact Reference Guide* also explains the concepts and facilities provided by VISION:Transact for the support of DB2 relational data bases.
- The *Advantage VISION:Transact Message Guide* contains messages and condition/ ABEND codes issued by the VISION:Transact generator, and error/trace messages and ABEND codes issued by the VISION:Transact generated applications, with an explanation of each. All messages issued by the VISION:Transact generator are prefixed with MK5.

How This Guide Is Organized

This guide consists of eight chapters and an appendix. It is organized to guide you through the use of the VISION:Transact system phase-by-phase.

- [Chapter 1, “Introduction”](#) gives a brief overview of this guide and the other VISION:Transact documentation, and describes the organization of this guide.
- [Chapter 2, “System Overview”](#) describes the VISION:Transact system in general terms.
- [Chapter 3, “Definition and Source Statement Retrieval Runs”](#) describes how to store file and screen processing definitions and requests and request groups, and how to retrieve them from the library.
- [Chapter 4, “Application Generation Run”](#) describes how to perform an application generation with VISION:Transact.
- [Chapter 5, “Application Installation”](#) shows how to install a VISION:Transact generated application at the target communications monitor.
- [Chapter 6, “Application Execution”](#) describes the online execution of a generated application and provides information on tracing the execution of a generated application.
- [Chapter 7, “Characteristics of the Generated Application”](#) describes some characteristics of the executing application.
- [Chapter 8, “Library Utilities”](#) explains how to use the VISION:Transact library utilities.
- [Appendix A, “Reading VISION:Transact Application Dumps”](#) contains information on locating VISION:Transact's application-related data areas (such as temporary fields, working storage, user flags, and data base segments) in a storage dump.

This Environment Guide is designed to meet the needs of the person using the VISION:Transact generator to create applications and the person using the VISION:Transact executor to install the generated applications at the target communications monitor. In an installation where application development and execution are performed at the same site, the guide leads you step-by-step through the development and installation of online applications with VISION:Transact. In a distributed environment, where application development and execution occur at two different sites (and possibly under different operating systems), [Chapter 2, “System Overview”](#), [Chapter 5, “Application Installation”](#), [Chapter 6, “Application Execution”](#), [Chapter 7, “Characteristics of the Generated Application”](#), and [Appendix A, “Reading VISION:Transact Application Dumps”](#) are of primary importance to the person responsible for application installation at the execution-only site.

The job control language (JCL) examples in this guide follow a standard format. Upper-case characters are used for specifications that are thought to be usable as-is. Lower-case characters are used for specifications, such as data set names and volume serial numbers, that are installation dependent in nature.

System Overview

Description

VISION:Transact is an implementation system that simplifies the development of online business applications. VISION:Transact generated applications execute under the control of either IMS TM or CICS TS. VISION:Transact enables you to build applications that process and update information in data bases. You will find that VISION:Transact is easy to learn, requires minimal programming effort, and little knowledge of teleprocessing, as compared with other implementation tools. VISION:Transact also provides an extensive trace facility to assist in the testing and debugging of applications.

Development

Your VISION:Transact generated application accesses IMS data bases via DL/I and DB2 relational data bases via SQL. In addition, VISION:Transact accesses VSAM files in the CICS TS environment via the standard CICS TS facilities. Under IMS TM, VISION:Transact supports Fast Path data base processing. Because of these access methods, certain naming conventions are required. These are discussed in [Chapter 3, “Definition and Source Statement Retrieval Runs”](#) and [Chapter 7, “Characteristics of the Generated Application”](#) of this guide, and in the *Advantage VISION:Transact Reference Guide*.

When building VISION:Transact applications, you can perform the following steps:

- Data definition.
- Screen or COMMAREA processing definition.
- Catalog requests and request groups.
- Application generation, which can also define data, and screens or COMMAREAs, and catalog individual requests.
- Application installation.

These steps are discussed below.

Data Definitions, Screen or COMMAREA Processing Definitions, and Cataloged Requests

You can use a VISION:Transact definition run to catalog the data structures, screens or COMMAREAs, and requests to be processed. These definitions and requests can then be used when you compile your application. A single definition run can define data or screens or both, and catalog requests and request groups. Only in a definition run are data definitions cataloged in the library. Also, request groups can only be cataloged in a definition run.

To store definitions or requests on the VISION:Transact library (or Common Library), you must define the following data sets (described in [Chapter 3, “Definition and Source Statement Retrieval Runs”](#)):

- The load library data sets (STEPLIB) containing the VISION:Transact generator.
- The VISION:Transact library data set (M5LIB) containing file and screen or COMMAREA processing definitions and cataloged requests.
- The input data set (M5INPUT) containing VISION:Transact file definition and/or screen or COMMAREA processing definition source statements, and requests or request groups to be cataloged.
- The output data set (M5LIST) containing the formatted source statement listing, glossaries for all definitions cataloged in that run, and diagnostic messages.
- The output data set (M5PUNCH) that receives the screen or COMMAREA formatting source statements created by VISION:Transact. It will contain either IMS TM Message Format Service (MFS) statements or CICS TS Basic Mapping Support (BMS) statements for screens.

Application Generation

The primary output from a VISION:Transact application generation run (APPGEN) is an object module for the application.

During application generation, the VISION:Transact generator processes the following data sets (discussed in [Chapter 4, “Application Generation Run”](#)):

- The load library data sets (STEPLIB) containing the VISION:Transact generator, COMLIB subsystem, and if relational tables are used, the DB2 modules.
- The VISION:Transact library data set (M5LIB) containing file and screen or COMMAREA processing definitions, and any cataloged requests used in the VISION:Transact generated application.

- The input data set (M5INPUT) containing VISION:Transact source statements that define your application's specifications that may also include screen or COMMAREA definitions and requests to catalog on the library, and temporary file definitions.
- The output data set (M5LIST) containing the formatted source statement listing and diagnostic messages, and the recommended PSB generation statements and TP monitor interface statements.
- The output data set (M5PUNCH) containing the generated object modules and link edit control statements.
- If screen definitions are included in the input data set, you need the output data set (M5MAP) containing MFS or BMS source statements.
- If relational data bases (DB2) are used by the application, you need the output data set (M5PREP) containing the SQL statements to access the relational data bases.
- M5TPTBL is an optional output data set to contain recommended TP monitor and PSB generation statements.
- The output dataset (M5COMMAR) containing the COBOL statement copy book statements defining the COMMAREA for a CICS TS application.

Application Installation

Install your application by using the following steps (discussed in [Chapter 5, "Application Installation"](#)):

- Process the screen formatting source statements (BMS or MFS) that were output to M5PUNCH during the definition run or to M5MAP during the APPGEN.
- Process the SQL statements that were output to M5PREP during the APPGEN.
- Link edit the object modules that were output to M5PUNCH during the APPGEN.
- Define the application to the target communications monitor using statements output to M5LIST or M5TPTBL during the APPGEN. For a CICS TS application, you must define all BMS mapsets to CICS TS. The BMS mapset entries required may come from separate definition runs that include screen definitions as well as from the APPGEN run.
- Use the M5COMMAR output to map the CICS COMMAREA for a non-terminal application program.

Executing Your Application

Your VISION:Transact generated application is executed exactly like an application developed using any other high level language. In IMS TM, users execute the application by requesting an input screen belonging to the application to be formatted at the terminal using the /FORMAT command or by keying in the transaction code with optional data on a cleared screen. In CICS TS, the user enters a transaction code with optional data on an input screen, or invoking the application program with COMMAREA data.

VISION:Transact applications can be generated with or without execution tracing. More details on application execution are in [Chapter 6, “Application Execution”](#) and [Chapter 7, “Characteristics of the Generated Application”](#).

Definition and Source Statement Retrieval Runs

Definition Run

When you use the VISION:Transact generator and COMLIB subsystem to create an application, data base and screen or COMMAREA processing definitions are required. In addition, your application can use requests for additional processing and/or data base updating. You can execute a VISION:Transact definition run to store file and screen or COMMAREA processing definitions and catalog individual requests and request groups in the VISION:Transact library.

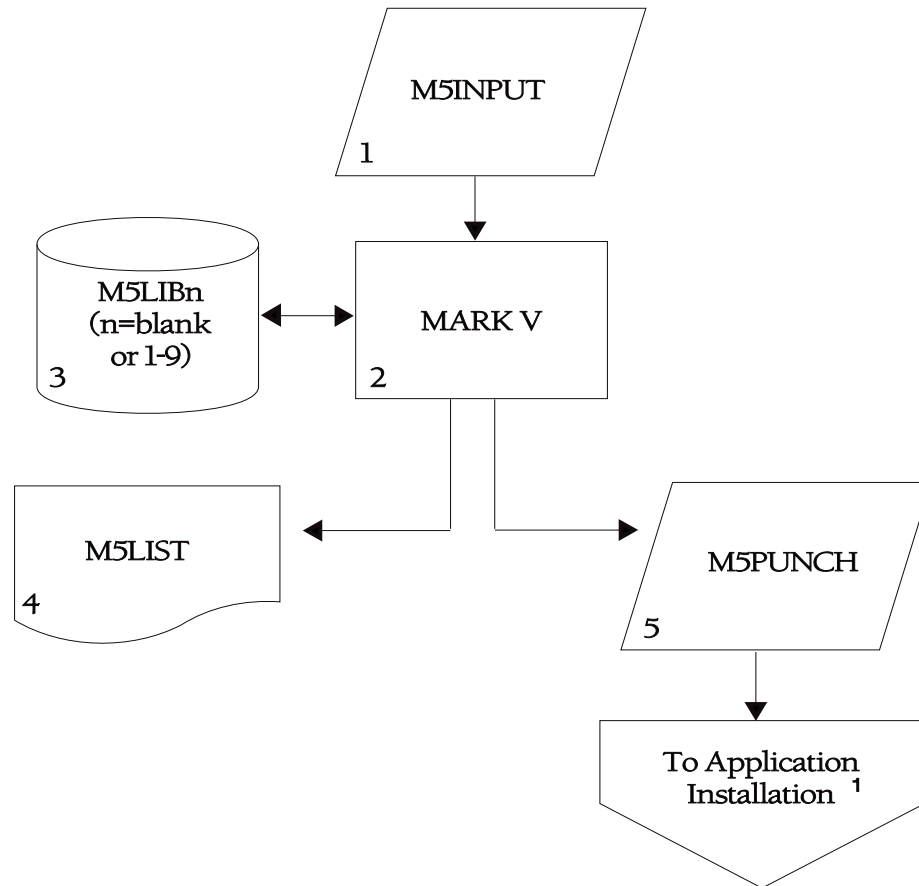
A definition run is also used to request glossaries (formatted listings) of cataloged file or screen definitions and listings of cataloged requests. Reports showing the names of all items cataloged on the VISION:Transact library or the names of all items of a given type (that is, screen or COMMAREA definition, file definition, or request) can be generated in a definition run through use of the CT statement.

VISION:Transact produces a glossary – a formatted listing – of any definition that is cataloged on the VISION:Transact library. In addition, VISION:Transact produces screen formatting source statements when screen processing definitions are cataloged on the VISION:Transact library. When you define screen definitions for IMS TM applications, VISION:Transact produces Message Format Service (MFS) statements. When you define screen definitions for CICS TS applications, VISION:Transact produces Basic Mapping Support (BMS) statements. The generated source statements are then input to the appropriate utilities when the application is installed.

The *Advantage VISION:Transact Reference Guide* contains a complete discussion of data base definition, and [Chapter 7, “Characteristics of the Generated Application”](#) of this guide discusses the processing of data bases by a VISION:Transact generated application. A complete discussion of screen processing definitions and requests can also be found in the *Advantage VISION:Transact Reference Guide*.

The flow and JCL for a VISION:Transact definition run is shown in [Figure 3-1](#). You can catalog file definitions for data bases, working storage, pseudo files, and PCB masks, as well as screen or COMMAREA processing definitions, requests, and request groups to the VISION:Transact library in a definition run.

Note: The M5PUNCH data set is required in a definition run.



Note:

```

//defrun JOB (acct-info)
2. //def EXEC PGM=MARKV,REGION=1024K
//STEPLIB DD DSN=markv.genlib,DISP=SHR
1. //M5INPUT DD DSN=markv.source,
// DISP=SHR
3. //M5LIB DD DSN=markv.m5lib,
// DISP=SHR
4. //M5LIST DD SYSOUT=*
5. //M5PUNCH DD DSN=markv.scrndef,
// DISP=(NEW,CATLG),
// UNIT=3350,
// VOL=SER=xxxxxx,
// SPACE=(CYL,(1,1))
//
  
```

Figure 3-1 Flow and JCL for a Definition Run

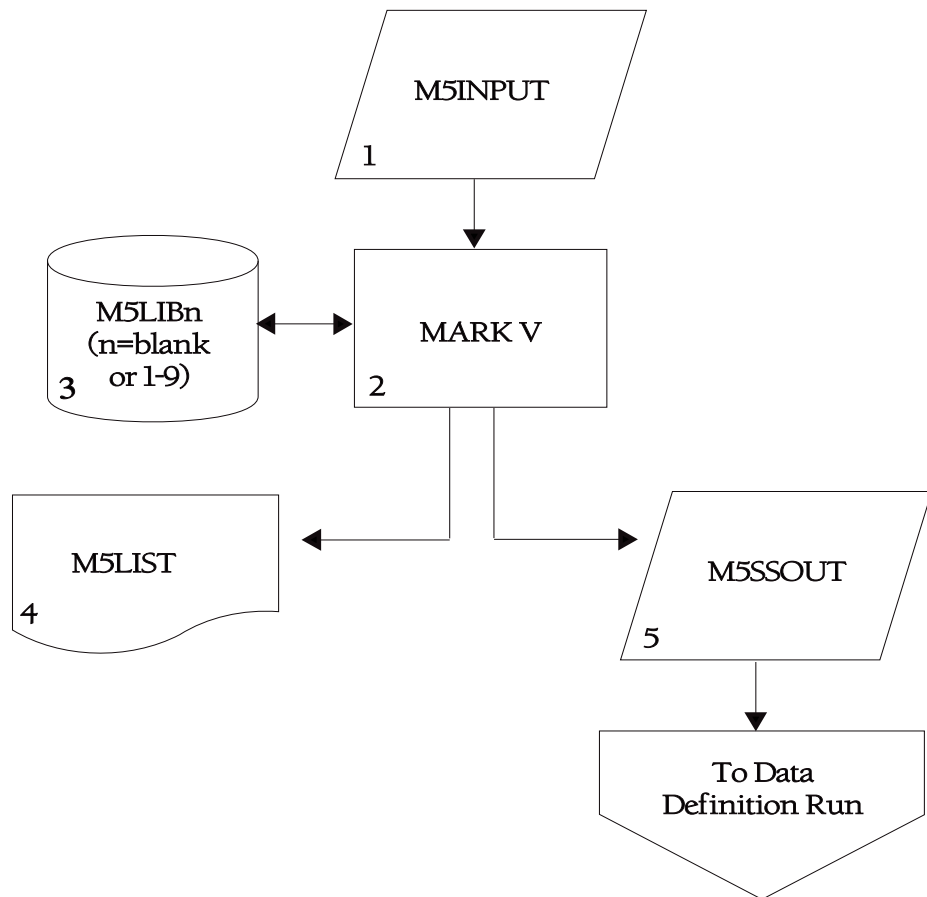
Notes	Explanation
1.	The M5INPUT data set consists of VISION:Transact source statements. The first source statement must be a Run Control (RC) statement specifying the run name, run type of D or blank (indicating a definition run), and a target TP monitor of I (IMS TM) or C (CICS TS). The RC statement can be directly followed by an RP statement that sequences the VISION:Transact libraries (M5LIBn) for cataloging and searching purposes. With no RP statement, all cataloging is to M5LIB. With an RP statement, all cataloging is to the first M5LIBn named by the RP statement. Next, any item tracking (IT) statements are input. Also CT statements can be input to list names or glossaries of items in the VISION:Transact library or to catalog or list request groups or list requests. The bulk of M5INPUT statements probably consist of file definition statements (FD, LS, LR, L0, and Ln), request statements (ER, TF, and PR), or screen definition statements (SD, SL, SS, S0, and Sn) or comment statements (AA).
2.	The library maintenance functions of the VISION:Transact generator are executed in a definition run. These functions process source statements from the M5INPUT data set, checking for syntax errors. If screen processing definitions are included in M5INPUT, BMS or MFS control statements are output to M5PUNCH. The execution of a definition run requires both the VISION:Transact generator and COMLIB subsystem load libraries.
3.	Definitions, requests, or request groups are cataloged or updated on the primary VISION:Transact library for later use in the generation of applications. Multiple M5LIBs are discussed in Multiple M5LIBs on page 3-6 .

Notes	Explanation
4.	The M5LIST output from a definition run includes a listing of the source statements in the M5INPUT data set, diagnostic messages, and glossaries of any definitions cataloged or updated on the VISION:Transact library (or requested for listing) in the definition run. When screen processing definitions are defined for a CICS TS application, this data set also lists the Processing Program Table (PPT) entry for the BMS mapset defined by control statements output to the M5PUNCH data set. A single VISION:Transact definition run creates one BMS mapset for all screens input in the run.
5.	When screen processing definitions are included in M5INPUT, the M5PUNCH output contains screen formatting source statements appropriate to the target communications monitor for which the screen processing definition is intended. When IMS TM is the target communications monitor, M5PUNCH consists of MFS source statements. When CICS TS is the target, M5PUNCH consists of BMS source statements. When you install the generated application, the M5PUNCH data set is input to the appropriate system utility to create a load module to be used by the generated application or target monitor. This further processing of the M5PUNCH data set is described in Chapter 5, “Application Installation” .

Note: The M5PUNCH DD statement is required in all definition runs. If a definition run includes only file definitions, requests, or request groups, M5PUNCH DD DUMMY can be specified.

Source Statement Retrieval Run

You can retrieve the source statements of items that are cataloged on the VISION:Transact library in a source statement retrieval run. [Figure 3-2](#) shows sample flow and JCL for source statement retrieval operations with VISION:Transact.

**Note:**

```

2. //retrieve JOB (acct-info)
//def EXEC PGM=MARKV,REGION=512K
//STEPLIB DD DSN=markv.genlib,DISP=SHR
1. //M5INPUT DD DSN=markv.source,
// DISP=SHR
3. //M5LIB DD DSN=markv.m5lib,
// DISP=SHR
4. //M5LIST DD SYSOUT=*
5. //M5SSOUT DD DSN=markv.retrieve,
// DISP=(NEW,CATLG),
// UNIT=3350,
// VOL=SER=xxxxxx,
// SPACE=(CYL,(1,1))
//

```

Figure 3-2 Flow and JCL for a Source Statement Retrieval Run

Notes	Explanation
1.	The M5INPUT data set consists of VISION:Transact source statements. The first source statement must be a Run Control (RC) statement specifying the run name and a run type of S which indicates a source statement retrieval run. The RC statement can be directly followed by an RP statement that sequences the VISION:Transact libraries (M5LIBn) for retrieval purposes. With no RP statement, M5LIB alone is searched if all items of a given type are wanted. With an RP statement, only the first library listed on it is searched for generic retrievals. When specific items are requested for retrieval, all available libraries are searched. The bulk of M5INPUT probably contains CT statements used to specify the items to be retrieved. Any other statements (including CT non-retrieval statements) are copied directly to M5SSOUT.
2.	The VISION:Transact library maintenance functions executed in a source statement retrieval run process the source statements from the M5INPUT data set, checking for syntax errors. The execution of a source statement retrieval run requires both the VISION:Transact generator and COMLIB subsystem load libraries.
3.	The VISION:Transact library is used for the retrieval of file definitions, screen definitions, requests, and request groups. Only the primary library is used for generic retrievals (see Multiple M5LIBs on page 3-6).
4.	The M5LIST output from a source statement retrieval run includes a listing of the source statements in the M5INPUT data set and diagnostic messages. If position [31] of the RC statement specifies an X, then a card-image listing of source statements output to M5SSOUT is produced.
5.	The M5SSOUT output from a source statement retrieval run is used to hold the card-image VISION:Transact source statements.

Multiple M5LIBs

Up to ten VISION:Transact libraries can be used in a definition or source statement retrieval run: M5LIB, M5LIB1, M5LIB2 . . . M5LIB9.

By default, M5LIB is the primary library. The primary library is the first searched for previously cataloged items and is the only library that is updated. The search continues with the default secondary libraries M5LIB1 to M5LIB9, whichever are present in the JCL, until the item is found. If duplicate items exist across libraries, the first matching item is used, stopping the search. You are not notified of any duplicates.

To override M5LIB as the primary library, code an RP statement with a parameter name of MULTILIB and a parameter value of M5LIB1 to M5LIB9. This redefines the primary library that is searched first and that may be updated. Use more RP statements each with a parameter name of MULTILIB and each naming a secondary library (not the primary library). The order in which these secondary libraries are named on RP statements defines the order in which they are searched. When RP MULTILIB statements are supplied, only those libraries named are searched.

In a definition run, the primary library is the only library into which items can be cataloged. When a CT listing operation is input in a definition run, the libraries that are searched depend on the operation. All libraries are searched, if the LFG, LSG, LFV, or DMP operations name specific items to be listed. When the LFG, LSG, LFV, or DMP operations select all items (no specific item is named), then only the primary library is searched. If the CT operation is LFN, LSN, LST, or LIN, only the primary library is searched.

In a source statement retrieval run, all libraries are searched when CT operators name specific items. Only the primary library is searched when CT operators request the source of all items of a given type (no item is named).

The primary library is always required, but you can use any combination of the secondary libraries. For example, using no RP statements, JCL with M5LIB and M5LIB6 is valid, while JCL with M5LIB1, M5LIB2, and not M5LIB is invalid.

The libraries can have different directory blocking factors, can be initialized by different releases, and can reside on different device types.

If additional M5LIBs are to be searched in a definition or source statement retrieval run, a type 0 message is output to M5LIST informing you that alternate libraries will be searched if necessary. If the type 0 message is not issued, the primary library mode is used; no alternate libraries are searched. Each time an item is used from a secondary library, another type 0 message is output to M5LIST stating the library in which the item was found.

Application Generation Run

Overview

In an application generation run (APPGEN), you input processing specifications to the VISION:Transact generator that produce an online application. The specifications include run control statements identifying the application, files, and whether or not execution tracing is to be included in the application; screen or COMMAREA section control statements describing screen-to-screen flow or COMMAREA section-to-COMMAREA section flow; and request or procedure statements describing data manipulation. You can also include data definitions and screen or COMMAREA processing definitions in the APPGEN. Using these specifications and any data and screen or COMMAREA processing definitions and requests and request groups cataloged in one or more definition runs, the VISION:Transact generator produces object code for the linkage editor.

If any RF statements reference relational data bases (record format S), the generator produces the required SQL statements to the M5PREP data set. This data set must be pre-processed and assembled, and the object code from the assembly must be included in the link edit for the application.

If any file definition is input in the APPGEN, two things are possible. If the file definition is already cataloged in a VISION:Transact library, the file definition statements input in the APPGEN are used to temporarily replace or update it. If the file definition does not exist on a VISION:Transact library, a temporary file definition is created for this APPGEN.

If screen or COMMAREA definitions are input in an APPGEN run, the generator catalogs them to the primary VISION:Transact library and produces BMS or MFS source statements to the M5MAP data set and COBOL data definition statements for the COMMAREA to the M5COMMAR dataset

If any requests are to be added, replaced, or deleted, the catalog action is performed using the primary VISION:Transact library. Request groups cannot be cataloged to the library in an APPGEN; a definition run must be used.

Note: If LIBOPNA=R is specified in M5PARM, screen definitions cannot be input and individual requests cannot be added, replaced, or deleted on the library in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement.

Application Generation Flow and JCL

[Figure 4-1](#) shows the flow of an APPGEN and the JCL required to run it when DB2 data bases are not referenced.

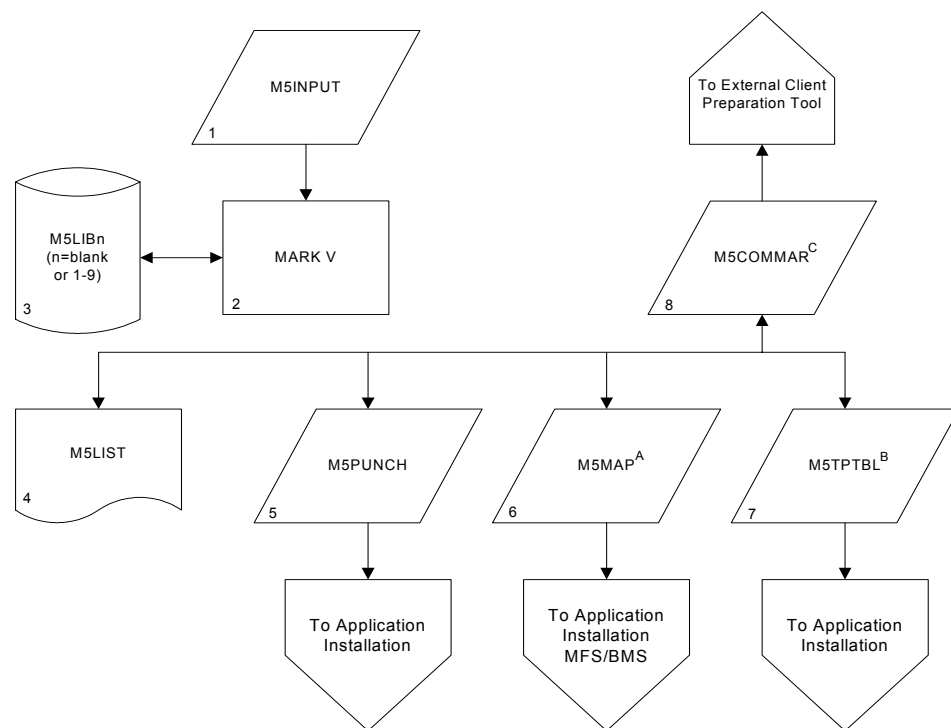


Figure 4-1 Flow and JCL for an Application Generation Run Without DB2 Data Bases (Page 1 of 2)

Note:

A. M5MAP data set is only needed if screen definitions are input in M5INPUT.

B. Optional data set containing TP monitor definitions.

C. Useful for External Client (EJB, etc.) Preparation.

```

2.      //appgen      JOB (acct-info)
        //GEN         EXEC PGM=MARKV,REGION=1024K
        //STEPLIB     DD DSN=markv.genlib,DISP=SHR
        //            DD DSN=comlib.loadlib,DISP=SHR
3.      //M5LIB       DD DSN=markv.m5lib,
        //            DISP=SHR
4.      //M5LIST      DD SYSOUT=*
5.      //M5PUNCH     DD DSN=markv.appgen,DISP=(NEW,CATLG),
        //            UNIT=SYSDA,
        //            VOL=SER=xxxxxx,
        //            SPACE=(CYL,(1,1))
1.      //M5INPUT     DD DSN=markv.source,DISP=SHR
6.      //M5MAP       DD DSN=markv.scrndef,DISP=(NEW,CATLG),
        //            UNIT=SYSDA,
        //            VOL=SER=xxxxxx,
        //            SPACE=(CYL,(1,1))
7.      //M5TPTBL     DD DSN=markv.table,DISP=(NEW,CATLG),
        //            UNIT=SYSDA,
        //            VOL=SER=xxxxxx,
        //            SPACE=(TRK,(5,1))
8.      //M5COMMAR    DD DSN=markv.commarea,DISP=(NEW,CATLG)
        //            UNIT=SYSDA
        //            VOL=SER=XXXXX,
        //            SPACE=(TRK,(5,1))

```

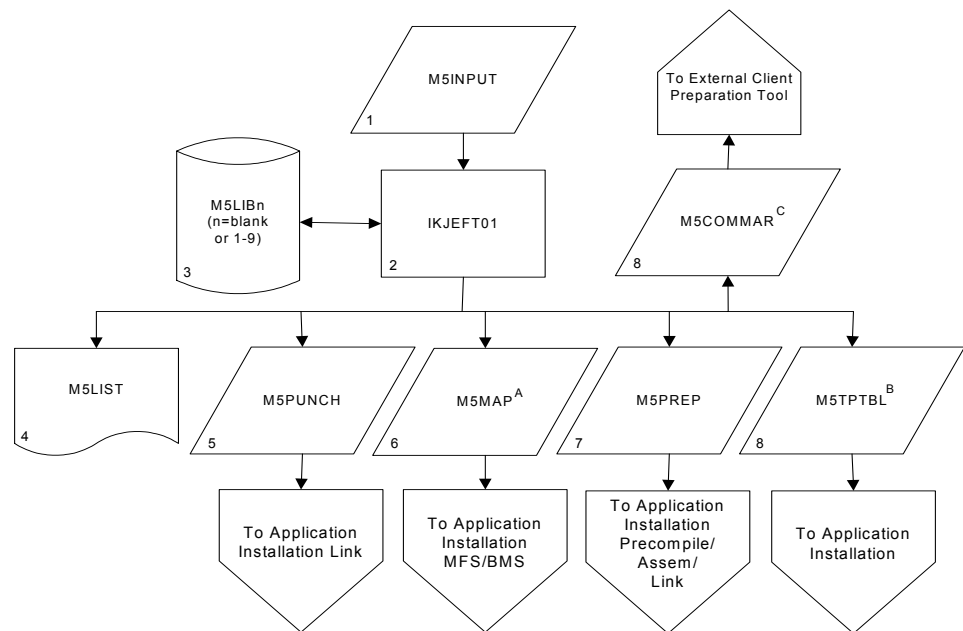
Figure 4-1 Flow and JCL for an Application Generation Run Without DB2 Data Bases (Page 2 of 2)

Notes	Explanation
1.	The M5INPUT data set contains VISION:Transact input statements. ■ The Run Control Group, which must appear first consists of: - The Run Control (RC) statement (Run Type of G for APPGEN). Note: Required statement. - Run Files (RF) statements. - Run Parameter (RP) statements. - Execution Tracing (?T) statement. - Item Tracking (IT) statements. ■ Data Definition (FD, LS, L0, Ln) statements that temporarily create, replace, or update file definitions. ■ Temporary Field (TF) statements. ■ Screen or COMMAREA Control (SC) statements.¹ Note: Required statement. ■ Screen or COMMAREA Definition (SD, SL, SS, S0, Sn) statements that are cataloged in the primary library. Note: If LIBOPNA=R is specified in M5PARM, screen or COMMAREA definition statements cannot be input and requests cannot be added, replaced, or deleted on the library in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement. ■ Request (ER and PR) statements, grouped by request name, including TF statements. Note: If LIBOPNA=R is specified in M5PARM, screen or COMMAREA definition statements cannot be input and requests cannot be added, replaced, or deleted on the library in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement. ■ Procedure (ER and ASL) statements. ■ Cataloged Request Retrieval (CR) statements. Comment (AA) statements can appear anywhere in a statement set that has a run name or a request name. Comments can also appear in Screen or COMMAREA Control statement sets. If Comment statements are included in a Screen or COMMAREA Control set, positions [1-8] must be blank. If included in a data or screen or COMMAREA definition statement set, they print in the source listing but not in the glossary.

Notes	Explanation
2.	The VISION:Transact generator is a batch compiler. It checks the syntax of your input statements from M5INPUT. If the statements are valid, it generates object code and outputs it to M5PUNCH. The execution of an APPGEN run requires both the VISION:Transact generator and COMLIB subsystem load libraries.
3.	The M5LIB data set is the VISION:Transact library, containing cataloged data base definitions, screen or COMMAREA processing definitions, and, possibly, cataloged requests. Multiple M5LIBs are discussed in Multiple M5LIBs on page 4-12 .
4.	The M5LIST data set contains the following listings, which are produced as part of the application generation process: ■ The formatted source statements, diagnostic messages, and definition glossaries. ■ The PSB generation statements. If a PSB is required by the generated application, VISION:Transact determines the minimum PSB requirements for the application. The generated data base PCBs assume a physical DBD without logical relationships. If LCHILD specifications exist in the DBD, the PSB recommendations regarding KEYLEN may require modification. VISION:Transact recommends a PROCOPT of AP for any data base being updated, but a more restrictive specification is acceptable when it is consistent with segment actions specified in screen processing definitions. In addition, for IMS TM applications, an alternate PCB is generated for the destination of error messages if the application encounters a critical error. (PSB generation statements can be output to the M5TPTBL data set.) Chapter 7, “Characteristics of the Generated Application” has more information regarding PSBs. ■ The target communications monitor interface statements. VISION:Transact determines the minimum statements required to identify your application to the communications monitor for which it is generated. For IMS TM, APPLCTN and TRANSACT macro statements are listed. You must include these in the IMS system generation (IMSGEN). Only the required parameters are listed. Other specifications may be added as needed. For CICS TS, recommended Transaction, Program, and Mapset CSD DEFINE statements are listed. Again, these are only the required parameters, and others may be added. (These statements can be output to the M5TPTBL data set.) ■ The Design Manual, if requested via the RP MANUAL statement.

Notes	Explanation
5.	The M5PUNCH data set contains an object module and linkage editor control statements. You link edit this object module as part of the application installation process (see Chapter 5, “Application Installation”).
6.	The M5MAP data set is only needed to output screen formatting statements when screen definitions are input to the APPGEN. Under IMS TM, M5MAP contains MFS source statements. Under CICS TS, M5MAP contains BMS source statements. Further processing of the M5MAP data set is described in Chapter 5, “Application Installation” .
7.	The M5TPTBL data set is optional. It contains VISION:Transact recommendations for TP monitor definitions. For IMS TM it contains the PSB, APPLCTN, and TRANSACT entries. For CICS it contains the Transaction, Program and Mapset CSD DEFINE statements for the generated program. Additionally, it contains DEFINE statements for all mapsets included in the APPGEN run; it does not include DEFINE statements for screens that were previously cataloged in definition runs.
8.	The M5COMMAR data set is optional. It contains COBOL data definition statements that define the layout of the COMMAREA whenever the VISION:Transact application program is not invoked via the 3270 interface.

[Figure 4-2](#) shows the flow of an APPGEN and the JCL required to run it when DB2 data bases are referenced.

**Note:**

- A. M5MAP data set is only needed if screen definitions are input in M5INPUT.
- B. Optional data set containing TP monitor definitions.
- C. Useful only for External Client (EJB, etc.) Preparation.

Figure 4-2 Flow and JCL for an Application Generation Run With DB2 Data Bases
(Page 1 of 2)

Note:**Notes:**

```

//appgen JOB (acct-info)
2. //GEN EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=1500K
//STEPLIB DD DSN=markv.genlib,DISP=SHR
// DD DSN=comlib.loadlib,DISP=SHR
// DD DSN=db2.dsnload,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
        DSN SYSTEM(ssid) where ssid = your DB2 subsystem id
        RUN PROGRAM(MARKV) PLAN(planname) LIB('markv.genlib')
        END
3. //M5LIB DD DSN=markv.m5lib,
// DISP=SHR
4. //M5LIST DD SYSOUT=*
5. //M5PUNCH DD DSN=markv.appgen,DISP=(NEW,CATLG),
// UNIT=SYSDA,
// VOL=SER=xxxxxx,
// SPACE=(CYL,(1,1))
1. //M5INPUT DD DSN=markv.source,DISP=SHR
6. //M5MAP DD DSN=markv.scrndef,DISP=(NEW,CATLG),
// UNIT=SYSDA,
// VOL=SER=xxxxxx,
// SPACE=(CYL,(1,1))
7. //M5PREP DD DSN=markv.sqlprep,DISP=(NEW,CATLG),
// UNIT=SYSDA,
// VOL=SER=xxxxxx,
// SPACE=(TRK,(5,2))
8. //M5TPTBL DD DSN=markv.table,DISP=(NEW,CATLG),
// UNIT=SYSDA,
// VOL=SER=xxxxxx,
// SPACE=(TRK,(5,1))
9. //M5COMMAR DD DSN=markv.commarea,DISP=(NEW,CATLG)
// UNIT=SYSDA
// VOL=SER=XXXXX,
// SPACE=(TRK,(5,1))

```

Figure 4-2 Flow and JCL for an Application Generation Run With DB2 Data Bases
(Page 2 of 2)

Notes	Explanation
1.	The M5INPUT data set contains VISION:Transact input statements. ■ The Run Control Group, which must appear first consists of: - The Run Control (RC) statement (Run Type of G for APPGEN). Note: Required statement. - Run Files (RF) statements. - Run Parameter (RP) statements. - Execution Tracing (?T) statement. - Item Tracking (IT) statements. ■ Data Definition (FD, LS, LR, L0, Ln) statements that temporarily create, replace, or update file definitions. ■ Temporary Field (TF) statements. ■ Screen or COMMAREA Control (SC) statements. Note: Required statement. ■ Screen or COMMAREA Definition (SD, SL, SS, S0, Sn) statements that are cataloged in the primary library. Note: If LIBOPNA=R is specified in M5PARM, screen or COMMAREA definition statements cannot be input and requests cannot be added, replaced, or deleted on the library in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement. ■ Request (ER and PR) statements, grouped by request name, including TF statements. Note: If LIBOPNA=R is specified in M5PARM, screen or COMMAREA definition statements cannot be input and requests cannot be added, replaced, or deleted on the library in an APPGEN run. This specification can be overridden by the Library Open entry [28] on the RC statement. ■ Procedure (ER and ASL) statements. ■ Cataloged Request Retrieval (CR) statements. Comment (AA) statements can appear anywhere in a statement set that has a run name or a request name. Comments can also appear in Screen or COMMAREA Control statement sets. If Comment statements are included in a Screen or COMMAREA Control set, positions [1-8] must be blank. If included in a data, screen, or COMMAREA definition statement set, they print in the source listing but not in the glossary.

Notes	Explanation
2.	The TSO Terminal Monitor program IKJEFT01 is used to invoke a batch TSO session. The DSN command processor is used to attach DB2 and then invoke the VISION:Transact generator. The PLAN name in the run statement must match the PLAN name assigned to the generator at installation time. The VISION:Transact generator is a batch compiler. It checks the syntax of your input statements from M5INPUT. If the statements are valid, it generates object code and outputs it to M5PUNCH and outputs the SQL source statements for accessing DB2 relational data bases to M5PREP. The execution of an APPGEN run using relational tables requires the VISION:Transact generator, COMLIB subsystem, and DB2 load libraries.
3.	The M5LIB data set is the VISION:Transact library, containing cataloged data base definitions, screen or COMMAREA processing definitions, and, possibly, cataloged requests. Multiple M5LIBs are discussed in Multiple M5LIBs on page 4-12.

Notes	Explanation
4.	The M5LIST data set contains the following listings, which are produced as part of the application generation process: ■ The formatted source statements, diagnostic messages, and definition glossaries. ■ The PSB generation statements. If a PSB is required by the generated application, VISION:Transact determines the minimum PSB requirements for the application. The generated data base PCBs assume a physical DBD without logical relationships. If LCHILD specifications exist in the DBD, the PSB recommendations regarding KEYLEN may require modification. VISION:Transact recommends a PROCOPT of AP for any data base being updated, but a more restrictive specification is acceptable when it is consistent with segment actions specified in screen processing definitions. In addition, for IMS TM applications, an alternate PCB is generated for the destination of error messages if the application encounters a critical error. (PSB generation statements can be output to the M5TPTBL data set.) Chapter 7, “Characteristics of the Generated Application” has more information regarding PSBs. ■ The target communications monitor interface statements. VISION:Transact determines the minimum statements required to identify your application to the communications monitor for which it is generated. For IMS TM, APPLCTN and TRANSACT macro statements are listed. You must include these in the IMS system generation (IMSGEN). Only the required parameters are listed. Other specifications may be added as needed. For CICS TS, recommended Transaction, Program, Mapset, and DB2 Entry CSD DEFINE statements are listed. Again, these are only the required parameters, and others can be added. (These statements can be output to the M5TPTBL data set.) ■ The Design Manual, if requested via the RP MANUAL statement.
5.	The M5PUNCH data set contains an application object module and linkage editor control statements. You link edit this object module and the SQL object module as part of the application installation process (see Chapter 5, “Application Installation”).
6.	The M5MAP data set is only needed to output screen formatting statements when screen definitions are input to the APPGEN. Under IMS TM, M5MAP contains MFS source statements. Under CICS TS, M5MAP contains BMS source statements. Further processing of the M5MAP data set is described in Chapter 5, “Application Installation”.

Notes	Explanation
7.	The M5PREP data set is needed to output the SQL source statements when relational data bases (Record Format S) are used by the application. The SQL source is precompiled and assembled, and input to the linkage editor along with the application object module output to M5PUNCH. Further processing of the M5PREP data set is described in Chapter 5, "Application Installation" .
8.	The M5TPTBL data set is optional. It contains VISION:Transact recommendations for TP monitor definitions. For IMS TM it contains the PSB, APPLCTN, and TRANSACT entries. For CICS TS it contains the Transaction, Program and Mapset CSD DEFINE statements for the generated program. Additionally, it contains a DEFINE statement for all Mapsets included in the APPGEN run; it does not include DEFINE statements for Mapsets that were previously cataloged in definition runs. If relational files are used, it also contains a recommended DEFINE DB2 Entry statement.
9.	The M5COMMAR data set is optional. It contains COBOL data definition statements that define the layout of the COMMAREA whenever the VISION:Transact application program is not invoked via the 3270 interface.

Multiple M5LIBs

Up to ten VISION:Transact libraries can be used in an application generation run: M5LIB, M5LIB1, M5LIB2 . . . M5LIB9.

By default, M5LIB is the primary library. The primary library is the first searched for previously cataloged items. The search continues with the default secondary libraries M5LIB1 to M5LIB9, whichever are present in the JCL, until the item is found. When searching for a request/request group specified on a CR statement, the search is first made for a request group with the specified name through all available libraries. Then, if the search fails for a request group, the search is made for a request with the specified name through the available libraries. If duplicate items exist across libraries, the first matching item is used, stopping the search. You are not notified of any duplicates.

To override M5LIB as the primary library, code an RP statement with a parameter name of MULTILIB and a parameter value of M5LIB1 to M5LIB9. This redefines the primary library that is searched first and that may be updated. Use more RP statements each with a parameter name of MULTILIB and each naming a secondary library (not the primary library). The order in which these secondary libraries are named on RP statements defines the order in which they are searched. When RP MULTILIB statements are supplied, only those libraries named are searched.

In an APPGEN run, the primary library is the only library into which screens and requests (specified for cataloging) are added, replaced, or deleted. The primary library can only be updated during an APPGEN run when the library is opened for update (see the LIBOPNA keyword in M5PARM and the Library Open entry [28] on the RC statement).

The primary library is always required, but you can use any combination of the secondary libraries. For example, using no RP statement, JCL with M5LIB and M5LIB6 is valid, while JCL with M5LIB1, M5LIB2, and not M5LIB is invalid.

The libraries can have different directory blocking factors, can be initialized by different releases, and can reside on different device types.

If additional M5LIBs are to be searched in an APPGEN, a type 0 message is output to M5LIST indicating that alternate libraries will be searched if necessary. If the type 0 message is not issued, the primary library mode is used; no alternate libraries are searched. Each time an item is used from a secondary library, another type 0 message is output to M5LIST stating the library in which the item was found.

Application Installation

Overview

Once you have generated an application with the VISION:Transact generator, you must install the application to execute it under the target communications monitor. The installation of the generated application includes:

- Processing the screen formatting source statements contained in the M5PUNCH data set that was output during the definition run, or the M5MAP data set output during the APPGEN.
- Precompiling and assembling the SQL statements contained in the M5PREP data set in the APPGEN when DB2 data bases are used.
- Link editing the generated application object module that was output to the M5PUNCH data set in the APPGEN. If DB2 data bases are used, the application object module output to M5PUNCH and the SQL object module output from precompiling and assembling the SQL source contained in M5PREP, are both link edited together.
- Identifying the application to the target communications monitor, using the names specified by VISION:Transact in the APPGEN run (M5LIST or M5TPTBL data set), and, for a CICS TS application, in the screen processing definition run.
- Generating appropriate PSBs and ACBs – paying heed to the minimum requirements noted by VISION:Transact in the APPGEN run (M5LIST and/or M5TPTBL data set) – if the installed application is to process IMS data bases or execute under IMS TM.
- Binding the application plan when DB2 data bases are used.

Processing Screen Formatting Source Statements

When you define screen processing definitions for your application, VISION:Transact produces an M5PUNCH data set in a definition run or an M5MAP data set in an APPGEN containing screen format source statements for the target communications monitor. If the target monitor is IMS TM, the screen format source statements are Message Format Service (MFS) utility control statements. If the target monitor is CICS TS, the screen format source statements are Basic Mapping Support (BMS) statements. These are processed by the appropriate utilities to enable the screen processing of the generated application.

IMS TM MFS Utility JCL

[Figure 5-1](#) shows the JCL for processing the MFS utility control statements produced by VISION:Transact. Executing the utilities in [Figure 5-1](#) stores control block descriptors in the IMS.FORMAT library for use by the online application.

```
//scrninst JOB (acct-info)
//JOB LIB DD DSN=IMS.RESLIB,DISP=SHR
//MFS1 EXEC PGM=DFSUPAAO,REGION=2M,
// PARM='NOXREF,NOCOMP,NOSUBS,NODIAG,NOCOMPRESS,STOPRC=8'
//SYSIN DD DSN=markv.scrnref,
// DISP=(OLD,DELETE)
//REFIN DD DSN=IMS.REFERAL,DISP=OLD
//REFOUT DD DSN=IMS.REFERAL,DISP=OLD
//REFRD DD DSN=IMS.REFERAL,DISP=OLD
//SYSTEXT DD DSN=&&TXTPASS,UNIT=SYSDA,
// SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//DUMMY DD DSN=IMSVS.PROCLIB(REFCPY),DISP=SHR
//UTPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SEQBLKS DD DSN=&&BLKS,DISP=(NEW,PASS),
// UNIT=SYSDA,SPACE=(CYL,(1,1))
//*
//MFS2 EXEC PGM=DFSUNUBO,REGION=2M,PARM='NOCOMPRESS',
// COND=(8,LT,MFS1)
//SEQBLKS DD DSN=&&BLKS,DISP=(OLD,DELETE)
//UTPRINT DD SYSOUT=*
//FORMAT DD DSN=IMS.FORMAT,DISP=OLD
//DUMMY DD DSN=IMS.PROCLIB(FMTCOPY),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(2,2))
//
```

Figure 5-1 JCL for Processing the MFS Control Statements

Note: Since steps MFS1 and MFS2 are controlled by IBM software, changes to the JCL are the responsibility of the customer.

CICS TS BMS Map Generation

[Figure 5-2](#) contains sample JCL for the assembly and link edit of the BMS utility control statements produced by VISION:Transact. These generate the physical maps required for online application execution. The VISION:Transact generated application provides its own symbolic descriptions of these physical maps, so there is no need for you to assemble DSECTs separately.

```
//scrninst JOB (acct-info)
//ASM EXEC PGM=ASMA90,REGION=2M,TIME=1,
//          PARM=(NODECK,OBJECT,'XREF(FULL)',NORENT,NORLD,
//          'LINECOUNT(50)', 'SYSPARM(OS)')
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=cics.maclib,DISP=SHR
//          DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIN DD UNIT=SYSDA,DISP=(NEW,PASS),SPACE=(TRK,(5,5)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200,DSORG=PS),DSN=&&OBJ
//SYSIN DD DSN=markv.scrndef,DISP=(OLD,DELETE)
//LNK EXEC PGM=IEWL,REGION=2M,COND=(0,NE,ASM),
//          PARM='LET,LIST,NCAL,RENT'
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DSN=user.bmslib(mapsetn),DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIB DD DSN=markv.cicslib,DISP=SHR
//SYSLIN DD DSN=&&OBJ,DISP=(SHR,DELETE)
//
```

Figure 5-2 JCL for Assembly and Link Edit of BMS Maps

Note:

1. BMS mapsets must be assembled as unaligned.
2. Since steps ASM and LNK are controlled by IBM software, changes to the JCL are the responsibility of the customer.

Application Link Edit

The M5PUNCH data set produced by VISION:Transact during the APPGEN contains an object module that can be link edited into an online load library.

[Figure 5-3](#) shows the JCL for the link edit of an application generated for the IMS TM target monitor when DB2 data bases are not used.

```
//appinst JOB (acct-info)
//LINK EXEC PGM=IEWL,REGION=2M,
// PARM='LET,LIST,RENT'
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DSN=applib,
// DISP=OLD
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIB DD DSN=markv.imslib,
// DISP=SHR
//IMSLIB DD DSN=IMS.RESLIB,DISP=SHR
//SYSLIN DD DSN=markv.appgen,
// DISP=(OLD,DELETE)
//
```

Figure 5-3 JCL for the Link Edit of an IMS TM Application Without DB2

[Figure 5-4](#) shows the JCL for the link edit of an application generated for the IMS TM environment when one or more DB2 data bases are used.

```
//appinst JOB (acct-info)
//SQLPREP EXEC DSNHASM,MEM=appname,USER=userid,
// PARM,PC='HOST(ASM),STDSQL(YES)',
// PARM,LKED='LET,LIST,REUS,XREF'
//PC.SYSLIB DD DUMMY
//PC.SYSIN DD DSN=markv.sqlprep,DISP=(OLD,DELETE)
//LKED.SYSLIB DD DSN=markv.imslib,DISP=SHR
// DD DSN=IMS.RESLIB,DISP=SHR
//LKED.SYSLMOD DD DSN=applib,DISP=OLD
//LKED.SYSIN DD DSN=markv.appgen,DISP=(OLD,DELETE)
```

Figure 5-4 JCL for the Precompile, Assembly, and Link Edit of an IMS TM Application With DB2

In [Figure 5-3](#) and [Figure 5-4](#) the SYSLIB data set contains the VISION:Transact executor for IMS TM applications. This data set supplies a few common routines which are link edited into the generated application.

Note: Since the LINK and SQLPREP steps are controlled by IBM software, changes to the JCL are the responsibility of the customer. The SQLPREP step invokes the DSNHASM procedure which requires the user to have previously allocated a DBRMLIB partitioned data set. The DBRMLIB PDS has a name in the form of “userid.DBRMLIB.DATA” and contains fixed length 80-byte records.

[Figure 5-5](#) shows the JCL for the link edit of an application generated for the CICS TS target monitor when DB2 data bases are not used.

```
//appinst JOB (acct-info)
//LINK EXEC PGM=IEWL,REGION=2M,
//      PARM='LET,LIST,RENT'
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DSN=applib,
//      DISP=OLD
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIB DD DSN=markv.cicslib,DISP=SHR
//      DD DSN=cics.loadlib,DISP=SHR
//SYSLIN DD DSN=markv.appgen,
//      DISP=(OLD,DELETE)
//
```

Figure 5-5 JCL for the Link Edit of a CICS TS Application Without DB2

[Figure 5-6](#) shows the JCL for the link edit of an application generated for the CICS TS environment when one or more DB2 data bases are used.

```
//appinst JOB (acct-info)
//SQLPREP EXEC DSNHASM,MEM=appname,USER=userid,
//      PARM.PC='HOST(ASM),STDSQL(YES)',
//      PARM.LKED='LET,LIST,RENT'
//PC.SYSLIB DD DUMMY
//PC.SYSIN DD DSN=markv.sqlprep,DISP=(OLD,DELETE)
//LKED.SYSLIB DD DSN=markv.cicslib,DISP=SHR
//      DD DSN=cics.loadlib,DISP=SHR
//      DD DSN=db2.dsnload,DISP=SHR
//LKED.SYSLMOD DD DSN=applib,DISP=OLD
//LKED.SYSIN DD DSN=markv.appgen,DISP=(OLD,DELETE)
```

Figure 5-6 JCL for the Precompile, Assembly, and Link Edit of a CICS TS Application With DB2

In [Figure 5-5](#) and [Figure 5-6](#) the SYSLIB data set contains the VISION:Transact executor for CICS TS applications. This data set supplies a few commonly used routines which are link edited into the generated application.

Note: Since the LINK and SQLPREP steps are controlled by IBM software, changes to the JCL are the responsibility of the customer. The SQLPREP step invokes the DSNHASM procedure which requires the user to have previously allocated a DBRMLIB partitioned data set. The DBRMLIB PDS has a name in the form of “userid.DBRMLIB.DATA” and contains fixed length 80-byte records.

Defining the Generated Application to the Target Communications Monitor

Identifying your generated application to the target communications monitor requires that you perform the standard definition process used to define any application to that monitor.

Defining an application to IMS TM consists of adding the application control (APPLCTN) and transaction (TRANSACTION) statements to the IMS system generation, and generating the appropriate program specification block (PSB) and application control block (ACB).

Defining an application to CICS TS consists of updating the CSD for the transaction program and BMS mapsets, and, if the application is to process IMS data bases, generating a usable PSB and ACB for the application and, if DB2 data bases are used, the CSD must be updated with the DB2 ENTRY definition.

When a CICS TS application is generated with execution tracing, the CICS TS CSD may need to be updated. If CICS TS transient data is to be used as a destination for trace output, then the CSD must be updated to define the transient data queue name used for tracing. Unless modified by your installation, all VISION:Transact applications use M5ET as the transient data queue name for trace output. For more information on execution tracing under CICS, see [Execution Tracing of CICS TS Applications on page 6-7](#).

The M5LIST output from the APPGEN indicates the minimum requirements for these control statements for the intended target monitor. In addition, you can request that the APPGEN produce a separate data set, M5TPTBL, containing these recommendations. In many cases you can tailor these minimum requirements for your online environment to meet particular installation standards. For CICS TS applications, the M5LIST output from a definition run including screen processing definitions contains the CSD entry for the mapset which includes all the screens defined in the run. All BMS mapsets must be defined to CICS TS in the CSD.

Note: For CICS TS the M5TPTBL file output during APPGEN does not include CSD entries for BMS mapsets (screens) defined in previous definition runs. The output to M5LIST in a definition run that includes screens must be included in the CICS TS CSD.

IMS TM APPLCTN and TRANSACT Macro Generation

[Figure 5-7](#) shows the process used to define the VISION:Transact generated application to IMS TM. The figure shows the M5LIST data set output from an APPGEN, which contains the APPLCTN and TRANSACT macro statement requirements. These statements are also output to the M5TPTBL file if it is assigned in the JCL. These statements (either from M5TPTBL or keyed in from M5LIST) must be input to an IMSGEN. In some installations, the IMSGEN is prepared by the data base administrator (DBA). VISION:Transact also outputs minimum PSB requirements to the M5LIST data set and if specified, to the M5TPTBL data set.

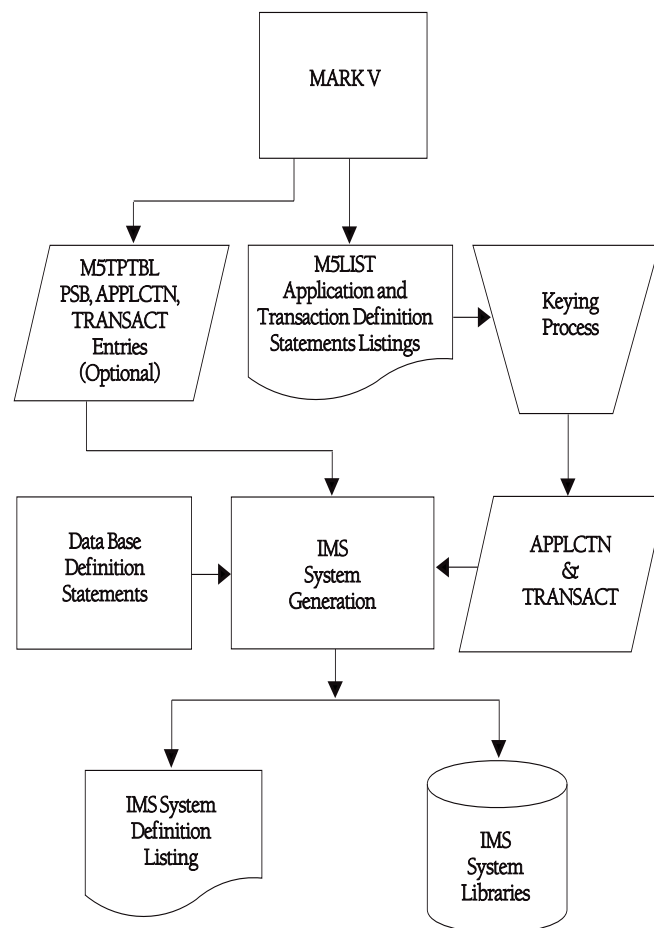


Figure 5-7 Generating APPLCTN and TRANSACT Statements

The M5LIST data set output from the APPGEN in [Figure 5-7](#) contains system generation macro statements like those shown in [Figure 5-8](#). These statements are subject to additional IMS specifications to meet installation standards, but the entries printed on the VISION:Transact listing are the minimum to be used. This is the same information placed in the M5TPTBL data set.

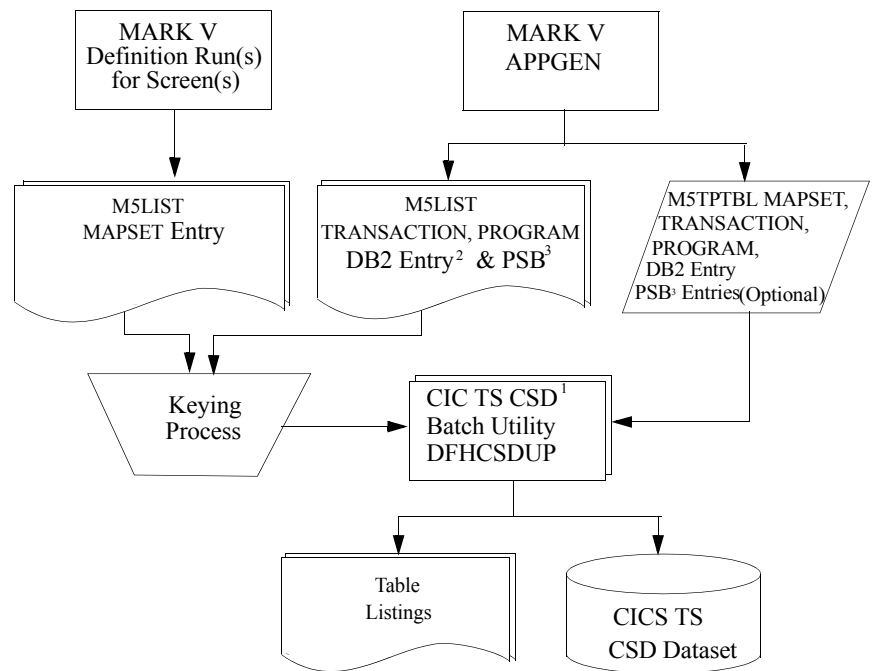
```
** M5TJ01  TYPE 0  THE IMS SYSTEM DEFINITION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -  
    APPLCTN  PSB=M14CSIT  
    TRANSACT MSGTYPE=SINGLESEG,MODE=SINGL, CODE=M510  
** M5TJ03  TYPE 0  THE IMS PSB GENERATION FOR THIS APPLICATION WILL REQUIRE THE FOLLOWING STATEMENTS -  
    PCB  TYPE=TP,EXPRESS=YES,LITERA=MASTER  
    PCB  TYPE=DB,POS=M,KEYLEN=0017,PROCOPT=AP,DBDNAME=CUSTOMDB  
    SENSEG NAME=CUSTOMER  
    SENSEG NAME=ORDER,PARENT=CUSTOMER  
    SENSEG NAME=ITEMORD,PARENT=ORDER  
    PCB  TYPE=DB,POS=M,KEYLEN=0014,PROCOPT=AP,DBDNAME=ITEMDB  
    SENSEG NAME=ITEMROOT  
    SENSEG NAME=BKORDERS,PARENT=ITEMROOT  
    PCB  TYPE=DB,POS=M,KEYLEN=0007,PROCOPT=GP,DBDNAME=ITEMDB  
    SENSEG NAME=ITEMROOT  
    PSBGEN LANG=ASSEM,MAXQ=02,PSBNAME=M14CSIT  
    END
```

Figure 5-8 An Example of the Recommended APPLCTN, TRANSACT, and PSB Requirements

CICS TS CSD Updates

[Figure 5-9](#) depicts the process used to define the VISION:Transact generated application to CICS TS. The figure shows that the M5LIST data set output from a screen definition run contains the required CSD entry for the BMS mapset. The M5LIST from an APPGEN run includes the statements required to update the CSD. If IMS data bases are included in the application, M5LIST also contains the PSB requirements. These statements are also output to the M5TPTBL data set if it is assigned in the JCL. From either source (M5TPTBL or keyed from the M5LIST printout), these statements must be input to the CSD batch utility DFHCSDUP (CICS online resource definition may be used) or to the IMS PSBGEN as appropriate.

Note: When a CICS TS application is generated with execution tracing, the CSD may need to be updated. If CICS TS transient data is to be used as a destination for trace output, then the CSD must be updated to define the transient data queue name used for tracing. Unless modified by your installation, all VISION:Transact applications use M5ET as the transient data queue name for trace output. For more information on execution tracing under CICS, see [Execution Tracing of CICS TS Applications on page 6-7](#).

**Note:**

1. CICS online resource definition may also be used.
2. The DB2 ENTRY is used for DB2 applications only.
3. PSB source is input to the IMS PSBGEN and required if IMS data bases are used.

Figure 5-9 Updating the CICS TS CSD

The M5LIST data set output from the APPGEN in [Figure 5-9](#) contains CICS TS PCT, CSD, and RCT update statements and PSB source statements like those shown in [Figure 5-10](#). The PSB source statements are, of course, only produced if the application is to process IMS data bases. The RCT entry is only produced if your application uses DB2 data bases. These statements are subject to additional specifications to meet installation standards, but the entries printed on the VISION:Transact listing should be used as listed. This information is also contained in the M5TPTBL data set.

```

DEFINE GROUP (???) , DUMP (NO) , TWASIZE (00000) , TRANSACTION (M510) , *
    INDOUBT (BACKOUT) , PROGRAM (M510)
DEFINE GROUP (???) , LANGUAGE (ASSEMBLER) , PROGRAM (M510)
DEFINE DB2ENTRY (???) , GROUP (???) , TRANSID (M510) , PLAN (M510)

PCB TYPE=DB, POS=M, KEYLEN=0017, PROCOPT=AP, DBDNAME=CUSTOMDB
    SENSEG NAME=CUSTOMER
    SENSEG NAME=ORDER, PARENT=CUSTOMER
    SENSEG NAME=ITEMORD, PARENT=ORDER
PCB TYPE=DB, POS=M, KEYLEN=0014, PROCOPT=AP, DBDNAME=ITEMDB
    SENSEG NAME=ITEMROOT
    SENSEG NAME=BKORDERS, PARENT=ITEMROOT
PCB TYPE=DB, POS=M, KEYLEN=0007, PROCOPT=GP, DBDNAME=ITEMDB
    SENSEG NAME=ITEMROOT
PSBGEN LANG=ASSEM, MAXQ=002, PSBNAME=M14CSIT
END

```

Figure 5-10 An Example of the Recommended PCT, CSD, RCT, and PSB Requirements

The DEFINE PROGRAM is always required. However, the DEFINE TRANSACTION is not required if the program is invoked by another means, such as a Distributed Program Link. Additional parameters may be specified for these entries, as defined by company standards.

The DEFINE DB2ENTRY listed by VISION:Transact contains the required TRANS ID and PLAN parameters. The PLAN name recommended by VISION:Transact is the name of your application load module. This name must match the PLAN name specified in the BIND (see [Binding Your Application For DB2 on page 5-12](#)). The AUTH ID parameter is not included on the recommended RCT output by VISION:Transact as it is installation dependent and has an installation default value if omitted. The AUTH ID parameter determines the authorization ID that is to be used by DB2 when your application is executing under CICS. The authorization ID which is used can be either your CICS user ID, operator ID, terminal ID, or transaction ID; the authorization ID used must have execution privileges for the VISION:Transact application plan (see [Binding Your Application For DB2 on page 5-12](#)). Additional parameters may be specified on the DEFINE DB2ENTRY according to application requirements or company standards

By default, all VISION:Transact generated CICS applications issue a ROLLBACK sync point call during any error cycle (buffer reshow). To have CICS perform the file backout, the appropriate CICS TS startup, program , and file specifications must be set.

Cataloging screen processing definitions for a CICS TS-targeted application in a definition run produces a listing (on M5LIST) of the CSD entry for the BMS mapset generated for all screens in the run. Cataloging screen processing definitions for a CICS TS targeted application in an APPGEN run causes an additional CSD entry to be output to M5LIST and M5TPTBL for the BMS mapset. CSD entries for BMS mapsets generated during a definition or APPGEN run contain mapset name and information. Additional parameters may be specified, according to company standards.

PSB Generation

Most of the PCBs you use to run VISION:Transact generated applications should be taken directly from APPGEN recommendations. Logical data base PCBs and those for secondary indexes require modifications. For IMS TM only, you can modify the TP PCB recommendations for routing critical error messages or to add an alternate destination terminal.

For IMS TM, to determine where critical error messages are routed (the error screen), the value of the ERRDEST parameter in M5PARM is used.

If the ERRDEST parameter names an LTERM, the PSB must include a PCB (TYPE=TP) that names the same LTERM, and this PCB must specify EXPRESS=YES.

If the ERRDEST parameter does not name an LTERM, the first PCB you code (after the I/O PCB provided by IMS TM) is used for critical error messages, and it must specify EXPRESS=YES, and

- if it is not modifiable, it must name an LTERM as the destination for critical error messages,
- or if it is modifiable (MODIFY=YES), it must name no LTERM and critical error messages are sent to the originating terminal.

If you plan to use any alternate destinations for output messages under IMS TM (operand C of screen control is non-blank), use one additional PCB (not the ERRDEST PCB) that:

- names no LTERM,
- specifies EXPRESS=NO (default),
- and specifies MODIFY=YES.

Examples of PSB recommendations provided by VISION:Transact are shown in [Figure 5-8](#) (IMS TM) and [Figure 5-10](#) (CICS TS).

The TYPE=DB PCBs listed in [Figure 5-8](#) and [Figure 5-10](#) are required for these applications. These PCBs reflect the usage of IMS data bases by the application, and they specify the total length of the segment keys used to retrieve segments. The specifications POS=M and the P in PROCOPT are required. If your application must access data bases procedurally, you should add PCBs to these PSBs. The additional PCBs are not used by the generated application for its automatic data base access. For more information on how the application processes IMS data bases, refer to [Chapter 7, “Characteristics of the Generated Application”](#).

ACB Generation

Executing an online application that accesses IMS data bases requires the availability of an application control block (ACB). [Figure 5-11](#) diagrams the generation of an ACB, which combines input from the PSB and DBD libraries using the ACB generation utility. JCL for the utility can be found in the IBM IMS Utilities Manual.

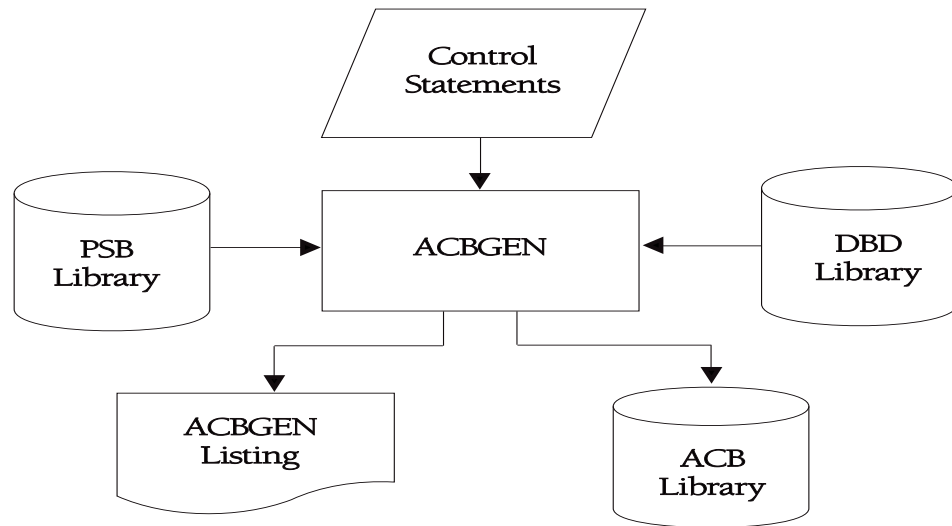


Figure 5-11 Generating an Application Control Block

Binding Your Application For DB2

The BIND process is only required if your application uses one or more DB2 data bases. The precompile, assembly, and link edit of your application described in [Application Link Edit on page 5-4](#) creates the input for the BIND process. [Figure 5-12](#) shows the JCL for binding an application for DB2.

```

//bind      JOB (acct-info)
//BIND      EXEC PGM=IKJEFT01,DYNAMMBR=20,REGION=1024K
//STEPLIB   DD DSN=db2.dsnload,DISP=SHR
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
DSN SYSTEM(ssid)      where ssid is your DB2 subsystem id
  BIND PLAN (planname) -
    LIBRARY ('userid.DBRMLIB.DATA') -
    MEMBER (appname) -
    ACTION (REPLACE) RETAIN -
    ISOLATION (RR)
  END
//
  
```

Figure 5-12 JCL to BIND an Application for DB2

The MEMBER name parameter for the BIND is the name of the data base request module (DBRM) output from the DB2 precompile. We recommend using your VISION:Transact application load module name for the parameter. The MEMBER name value specified for the BIND must match the MEM= value supplied for the DSNHASM procedure ([Figure 5-4](#) and [Figure 5-6](#)).

The PLAN name parameter for the BIND supplies the name of the application plan.

Important! *We recommend specifying the VISION:Transact application load module name as the plan name.*

Under IMS TM, if the application load module name differs from the plan name, an entry must be provided in the Resource Translation Table (RTT).

Alternatively, the BIND function may be performed online using DB2I. The parameters specified on the panels would be identical to those indicated in the sample job stream of [Figure 5-12](#).

After the BIND process has been successfully completed, all users who will be authorized to use the application must be GRANTED execution privileges for the plan created during the application installation process. After completing these steps, you can begin using the VISION:Transact application to access your DB2 data bases.

Application Execution

Overview

Once your VISION:Transact application is successfully installed, you can execute the application. [Application Execution Under IMS TM on page 6-1](#) describes application execution under IMS TM, and [Application Execution Under CICS TS on page 6-5](#) describes application execution under CICS TS. Additional sections describe the types of messages to expect during application execution, and VISION:Transact terminal support.

Application Execution Under IMS TM

[Figure 6-1](#) illustrates the typical flow of a VISION:Transact generated application executing in the IMS TM environment.

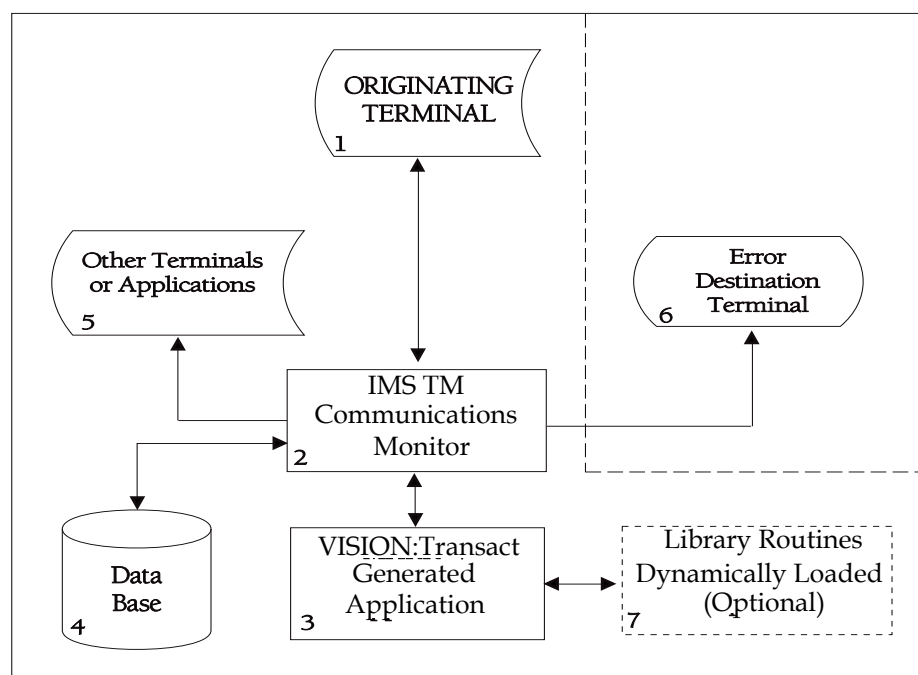


Figure 6-1 Application Execution Flow Under IMS TM System

Notes	Explanation
1.	For initial entry to a VISION:Transact application, the user can enter either the IMS/FORMAT command or the VISION:Transact transaction code. The IMS /FORMAT command causes IMS to output the formatted screen for the requested MOD name. Entering the VISION:Transact transaction code, which can optionally be followed by a screen name and/or data, causes IMS to place this data on the message queue and schedule the appropriate VISION:Transact application when an MP region is available. The VISION:Transact application then outputs an initial screen. This initial screen contains literals only or literals and data values. For more information on initial entry processing, see Chapter 7, "Characteristics of the Generated Application" . When the initial screen of the application is displayed, the user keys in the desired input data and presses the Enter key or a program function (PF) key to send the message to IMS.
2.	MFS maps the input screen using the DIF/DOF associated with the screen identifier into the message using the MID. IMS TM selects the edited message from the message queue and loads your application from the load library when a message processing (MP) region is available. Your application then gets control.

Notes	Explanation
3.	Your VISION:Transact generated application reads and validates the input screen data. If errors are detected, the input screen is redisplayed with an appropriate diagnostic message (see Messages Issued During Application Execution on page 6-10).
4.	The application issues DL/I or SQL calls to access and update IMS or DB2 data bases. If processing is successful, the specified output screen is selected and displayed on the appropriate terminal(s).
5.	You are able to specify output to terminals other than the originating terminal as well as other applications when generating the application.
6.	Through an EXPRESS PCB, serious application errors are displayed to an error destination, which can be an alternate terminal (i.e., the master terminal) or the originating terminal (see PSB Generation on page 5-11).
7.	The VISION:Transact installation parameters module (M5PARM) determines if the library routines are included in every application or dynamically loaded during online execution (XLOAD keyword). Dynamic loading allows the library routines to be preloaded into the MP region or placed in the LPA.

Additional IMS TM Commands

IMS provides additional commands for processing conversational applications. These commands are:

- /EXIT – Terminates a conversational application.
- /HOLD – Suspends temporarily a conversation between interactions.
- /RELEASE – Resumes a suspended conversation.

VISION:Transact Generated Applications and Existing IMS TM Applications and Subroutines

You can invoke other IMS TM applications (not necessarily generated by VISION:Transact) and non-VISION:Transact subroutines in the following ways:

- Include the name of the destination IMS TM application (VISION:Transact or non-VISION:Transact transaction code) in operand C of the screen control statement (see the *Advantage VISION:Transact Specifications Guide*). This is called a program-to-program message switch and transfers control to the new application.
- Use the TRANCODE flag on the screen. When the screen is input from the terminal, the contents of the TRANCODE flag determines the transaction to be invoked.

- In a request or procedure, include the name of the destination subroutine program as a parameter in a CALL statement or command (see the *Advantage VISION:Transact Reference Guide*). This is a subroutine CALL. The called routine returns control to the VISION:Transact generated application. The called routine cannot be a VISION:Transact application.

More information on message switching from a VISION:Transact generated application and calling user written subroutines is contained in [Chapter 7, “Characteristics of the Generated Application”](#).

Execution Tracing of IMS TM Applications

VISION:Transact applications can be generated to include or exclude execution tracing through the use of the ?T run control statement in an application generation (APPGEN) run.

When execution tracing is included in an IMS TM application (Trace Control entry of 0 or 1 on the ?T statement), the application must be executed under IMS TM BTS (batch or TSO) in order to obtain the trace output. Trace output is written to the BTSOUT data set. If BTS TSO is used, trace output is also written to the TSO terminal.

The trace output written to the BTSOUT data set or TSO terminal depends on the trace options specified. Trace options can be specified during APPGEN via the ?T statement or selected during execution through the use of the trace flags. More information on specifying trace options can be found in the *Advantage VISION:Transact Specifications Guide* and the *Advantage VISION:Transact Reference Guide*.

Some of the trace output generated by VISION:Transact can also be generated by BTS which can result in a duplication of output data. The TP calls and I/O areas for screens and saved working storage/pseudo files are output by VISION:Transact when the application flow tracing level is 2. These items can also be output by BTS using the MSG and SPA parameters of the ./O BTS input command. In addition, data base calls and I/O areas can be output by VISION:Transact when the SSDB or DSDB I/O call tracing level is 1 or 2. Data base calls can also be output by BTS using the DLITRACE or SQLTRACE parameters of the ./T command or the DB, TSODB, SQL, or TSOSQL parameters of the ./O command.

If it is desired, to eliminate the duplication of output under BTS, either BTS output or VISION:Transact trace should be disabled. The VISION:Transact trace can be more informative than the BTS output in certain areas such as the display of I/O areas for DL/I and SQL calls.

VISION:Transact determines whether an application generated with tracing is executing under IMS TM or BTS. All trace output is suppressed by VISION:Transact in the event an application generated with execution tracing is inadvertently executed under IMS TM.

If your installation does not use BTS to test IMS TM applications or your installation desires to redirect the trace output, please contact Technical Support for assistance.

Application Execution Under CICS TS

[Figure 6-2](#) illustrates the flow of a VISION:Transact application execution in the CICS TS environment. The numbers in the following description of the application flow refer to the numbers in the figure.

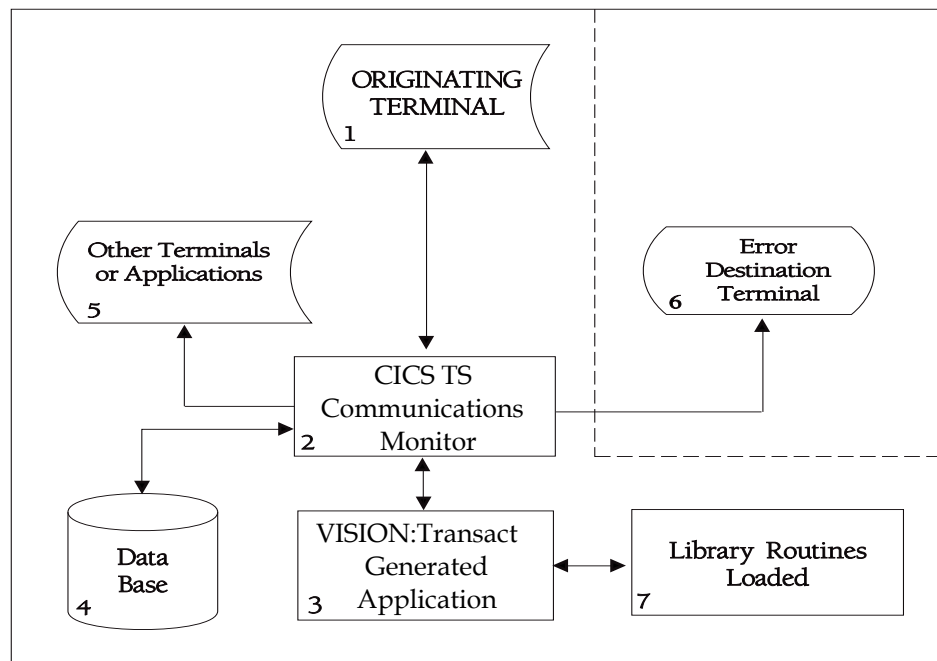


Figure 6-2 Application Execution Flow Under CICS TS System

For initial entry to the application, the user (1) enters the tran code (transaction identification) for the application which can optionally be followed by a screen name and/or data.

Note: For more information on initial entry to a VISION:Transact transaction, see [Chapter 7, “Characteristics of the Generated Application”](#).

CICS TS (2) validates the tran code, starts a task, and loads your application if not in storage. Your application (3) gets control, invokes BMS to output a formatted screen, creates a record in temporary storage to identify the program name and screen name for the next entry from that terminal ID, then exits, binding the terminal to the transaction if a tran code is coded on an SD statement or placed in the TRANCODE flag.

The user (1) enters data on the formatted screen and presses the Enter or PF key. CICS TS (2) starts a task again, loads your application if necessary, and the application retrieves the temporary storage record. The application (3) uses the temporary storage information to identify the screen last output and invokes BMS to map the input data according to the screen processing definition. Your application issues DL/I or SQL calls to access IMS or DB2 data bases, or invokes CICS TS file control for access to ISAM or VSAM files (4). After processing is complete, the application (3) again invokes BMS to format and write the selected output screen to the terminal, creates the temporary storage record to remember the name of the screen output, and exits. Your application has the ability to write to other terminals (5) during a transaction or start a new transaction.

Your VISION:Transact generated application terminates normally (deleting temporary storage records and not specifying TRANSID on its RETURN to CICS TS) under the following circumstances:

- When the selected output screen names no input transaction.
- When no output screen is named or selected by screen control for output to the originating terminal. In this case a clear screen is output to the terminal.
- When the Clear key or a PA key is pressed and the application sends a clear screen.

Note: The handling of the Clear and/or PA keys as a request for application termination is controlled on an installation basis in M5PARM and on an application basis on the RC statement.

The temporary storage records are also deleted if your application terminates abnormally.

If severe errors occur during the application, they can be routed to an error destination terminal (6) as well as the originating terminal.

The library routines (7) are used by the generated application to interface with CICS TS.

CICS TS Commands

The CICS TS CSMT or CEMT command must be used to bring in a new copy of your VISION:Transact BMS mapsets and programs (as well as the VISION:Transact library routines) when modifications have been made to them while the CICS TS system is up. The CSMT or CEMT command is also used to open files which have not been opened automatically by CICS TS during startup. The CICS TS CEDF command can be used for testing and debugging VISION:Transact generated applications interactively.

VISION:Transact Generated Applications and Existing CICS TS Applications and Subroutines

You can invoke other CICS TS programs (not necessarily generated by VISION:Transact) and non-VISION:Transact subroutines in the following ways:

- Place the transaction name of the CICS TS application (VISION:Transact or non-VISION:Transact) to be invoked into the TRANCODE flag and use the START parameter of the OC operator in screen control. This causes VISION:Transact to start up the transaction, passing it the output screen and any saved working storage and pseudo files.
- Before outputting a screen to the originating terminal, set the TRANCODE flag to the name of the CICS TS transaction that is to process the screen as input.
- In a request or a procedure, include the name of the destination subroutine program as a parameter of a CALL statement or command (see the *Advantage VISION:Transact Reference Guide*). This is a subroutine CALL. The called routine returns control to the VISION:Transact generated application.

Refer to [Chapter 7, “Characteristics of the Generated Application”](#) for more information on invoking CICS TS transactions and calling user written subroutines.

Execution Tracing of CICS TS Applications

VISION:Transact applications can be generated to include or exclude execution tracing through the use of the ?T run control statement in an application generation (APPGEN) run.

When execution tracing is included in a CICS TS application (Trace Control entry of 0 or 1 on the ?T statement), trace output can be written to three possible destinations under CICS TS. The CICS TS destinations used for trace output are the primary terminal, temporary storage, and/or transient data. These destinations can be enabled or disabled in any combination. The VISION:Transact CICS TS executor is delivered with all three destinations enabled. However, during the

installation of VISION:Transact, these destinations can be enabled or disabled based on your installation's requirements (see the *Advantage VISION:Transact Installation Guide* for details). Only temporary storage or a transient data queue may be used as trace output destinations when the application is running as a COMMAREA program.

The trace output written to these destinations depends on the trace options specified. Trace options can be specified during APPGEN via the ?T statement or selected during execution through the use of the trace flags. More information on specifying trace options can be found in the *Advantage VISION:Transact Specifications Guide* and *Advantage VISION:Transact Reference Guide*.

Trace output written to the primary terminal is displayed on the primary terminal while the application is executing, one screen page at a time. Trace output written to temporary storage and/or transient data can only be viewed *after* the application terminates and not while the application is executing. Temporary storage and transient data destinations are used to capture trace data for subsequent viewing in more detail or for hard copy output.

Trace output directed to the primary terminal begins displaying when application execution begins. When a screen's worth of trace data has been displayed, the application waits for input from the terminal. At this point, you can view the trace data before it is overwritten and then press the Enter or Clear key.

If the Enter key is pressed, the next screen's worth of trace data is displayed. If the Clear key is pressed, all trace output is suppressed for the duration of the interaction. Trace output is still written to temporary storage and/or transient data when the Clear key is pressed. If a key other than Enter or Clear is pressed, it is treated as the Enter key. It should be emphasized that once a key has been pressed, the trace data displayed is overwritten and no longer available for viewing unless it has also been written to temporary storage and/or transient data.

Application output screen(s) are displayed on the primary terminal at the end of the interaction cycle at which time input data can be entered for the next interaction. Application output screens routed to other terminals, such as a printer, are not displayed on the primary terminal. However, trace output informs you of the fact that they were written.

To view the temporary storage trace output, the CICS TS transaction CEBR can be used. The CEBR transaction allows you to browse, copy, and delete temporary storage queues and is documented in the IBM CICS TS Application Programming Guide. Unless modified by your installation, the eight-byte temporary storage queue name used for trace output is the literal M5ET followed by your four-byte terminal ID. It is the programmer's responsibility to delete (purge) his/her own trace temporary storage queue.

Trace records written to temporary storage are variable length with a maximum record size of 132. The trace records written to temporary storage are similar in content to those written to the primary terminal.

It is the programmer's responsibility for viewing and processing trace output written to transient data. Unless modified by your installation, the four byte transient data queue name used for trace output is M5ET. All VISION:Transact applications write trace data to the same transient data queue. Therefore, each transient data record contains the primary terminal ID as its first four bytes so that the trace records can be further identified.

The transient data queue name is defined to CICS TS by a TDQUEUE entry in the CSD. The CICS CSD entry for the transient data queue can be defined as desired to meet your installation's needs. For example, if you desire to use a batch job to print the trace output from your terminal, you can direct the trace output to an extrapartition transient data queue. Note that VISION:Transact automatically disables transient data as a trace destination if the transient data queue name is not defined to CICS TS (no error occurs). This allows installations to easily disable transient data as a trace destination by simply not defining the transient data queue name to CICS TS via an entry in the CSD.

Trace records written to transient data are variable length with a maximum record size of 137. The transient data queue must be defined as variable length in the CSD. The trace records written to transient data are similar in content to those written to the primary terminal. However, each record is prefixed by the four-byte primary (originating) terminal ID.

When trace records are written to temporary storage or transient data, any error that occurs on the write operation, with the exception of NOSPACE, suppresses subsequent writes to that destination. If the NOSPACE error occurs due to the insufficient space being available on the queue, then a message (prompt) is sent to the primary terminal informing you of the error. At this point, you may press PF2 to retry the write operation or press the Enter key to bypass the write operation as well as subsequent writes to the queue, and continue.

It should be noted that VISION:Transact writes the contents of the trace flags to CICS TS temporary storage at the end of each interaction cycle; this does not include recycle interactions. This is done to preserve the trace options in effect between interactions of a pseudo conversion. The eight-byte temporary storage queue name used to hold the contents of the trace flags is the literal M5TF followed by your four-byte terminal ID. The temporary storage record length is eight bytes. The record contains the value of each of the one-byte trace flags which are written in the following order: flag TCONTROL, TAPPFLOW, TPROC, TSEGACT, TILOOP, TDLOOP, TSSDBIO, and TDSDBIO.

Messages Issued During Application Execution

There are two types of messages that may be issued during VISION:Transact application execution: trace messages and error messages. The trace messages and error messages are contained in separate modules and are delivered in source to allow installations to tailor the messages if desired (see the *Advantage VISION:Transact Installation Guide* for details). The *Advantage VISION:Transact Message Guide* lists all trace and error messages issued during application execution along with an explanation of each message.

Trace messages are only output during application execution when the application has been generated with tracing and has enabled one or more execution tracing options. Trace messages are identified by a unique four-character code followed by the message text. The first two characters of a trace message identify the tracing option that triggered the output of the message, such as AF for application flow.

Error messages are output during application execution to identify input validation problems or serious problems for which execution of the application cannot continue. Error messages are identified by a unique three-character code followed by the message text. The generated application produces the following kinds of error messages:

- **Input Validation Messages**

These messages are displayed on the designated error line of the originating terminal when the input screen is re-displayed after an error condition is found. They are further divided into two types as follows:

1. Messages that are produced automatically by the VISION:Transact executor in response to error conditions related to the input data, such as validation criteria not met, numeric data overflow, data base segment not found, etc. The user responds by correcting the error and pressing the Enter or PF key to continue executing the application.

Note: You can override the text for these VISION:Transact messages with EC/ET operators in the screen processing definition.

2. Messages that you specify in the application by using the message sending (MS) operator in requests or screen processing definitions or MSG command in procedures.

- **Serious Error Messages**

Examples of serious errors are insufficient storage, a missing PCB, an unexpected DL/I status code, or an unexpected return code from CICS TS file control. In IMS TM, the error messages are routed to the originating terminal and displayed there in addition to the error destination terminal. However, if an ABEND occurs the messages are sent only to the error destination terminal specified in M5PARM or in the first user supplied PCB. In CICS TS, the error messages are displayed on the originating terminal as well as being routed to the error destination specified in M5PARM.

Terminal Support

VISION:Transact supports 3270-type terminals through MFS or BMS source statements generated in a definition or APPGEN run. The VISION:Transact generator depends on a terminal table defined in M5PARM to create the appropriate BMS or MFS source code. See the *Advantage VISION:Transact Installation Guide* for a discussion of these parameters.

COMMAREA Support

VISION:Transact supports CICS TS COMMAREA programs in addition to CICS Terminal Programs. A generated application can operate either as a COMMAREA program or a Terminal program. When a VISION:Transact application is started by CICS TS, the application determines whether a COMMAREA is available to the application. If a COMMAREA is available, the application operates as a COMMAREA program and the COMMAREA data becomes the input to the application and any output produced by the application will be returned to the caller via the COMMAREA. If a COMMAREA is not available to the application at startup the, the application operates as a Terminal program and the initial screen is displayed at the originating terminal

When a generated application operates as a COMMAREA program, the data in the COMMAREA is assumed to contain the input and output data for the application conforming to the layout described by the output directed to the M5COMMAR DD statement during the application generation run (see [Chapter 4, “Application Generation Run”](#)). A VISION:Transact application may be invoked as a COMMAREA program by any method available within CICS TS to invoke COMMAREA programs, including other VISION:Transact applications.

Characteristics of the Generated Application

This chapter describes the structure of your VISION:Transact generated application and how the application fits into your online environment.

Application Structure

The VISION:Transact generator uses a "building block" approach to the generation of object code, producing a separate CSECT for each logical unit of work. For example, a CSECT is generated for validation/segment action processing for each input screen in the application, and a CSECT for each request or procedure. CSECTs generated by VISION:Transact have names beginning with @@; these names may be the same from one application to the next, but the content and length of the CSECT varies according to the application's specifications. [Figure 7-1](#) lists and describes some of these generated CSECTs.

Functions that are common to all applications (such as initialization, PCB selection, ASMTDLI calls) are delivered pre-assembled to avoid redundant generation of object code. The generator includes link edit control statements within the M5PUNCH data set to control the inclusion of some of these pre-assembled routines when the application load module is built. The exact routines included in the link edit varies, based on the target communications monitor of the generated application.

The generator also includes a NAME link edit statement, using the program name from the RC statement. This assures that the name of the load module corresponds to the recommended APPLCTN/TRANSACT or CICS TS table update statements.

In any generated application the flow of control is the same. You can use [Figure 7-1](#) and your knowledge of VISION:Transact application flow to understand the control flow of CSECTs in your application. You can actually see the control of flow within an application if your application is generated with execution tracing and application flow tracing is enabled.

Generated For:	CSECT Name	Contents
Each Application	@@TEMPS	Code to initialize temporary and constant fields.
	@@WRKSTR	Code to initialize working storage fields.
	@@SCNTRL	Code to execute SC (screen control) statements.
	@@SCRTBL	Screen table: a table of vectors and constants used in processing input and output screens.
	@@COMPDT	Date-compiled character string in the form MMM DD,YYYY.
	@@COMPTM	Time-compiled character string in the form HH.MM.SS.
	@@FIVEPF	Copy of generator system modification record.
	@@Qvvvvv	Contains generated SQL statements and supporting code (DB2 relational data bases only).
		Note: The vvvvv represents an internally-generated sequence number used to avoid duplicate CSECT names.
	@@ESYMTB	Execution trace symbol locator table.
Each File	@@SG00F*	Segment table for file *.
	@@XB00F*	File usage information for file *.
	@@ESYMF*	Execution trace symbols (fields) for file *.
		Note: The * represents the file qualifier as assigned by the RF statement.
Each Screen	@@SG00mm	Segment table for screen mm.
	@@T#00mm	Request table for screen mm.
	@@SF00mm	Screen field location and attribute table for screen mm.
	@@ESYMmm	Execution trace symbols (fields) for screen mm.
		Note: The mm represents an internally-generated screen number.

Figure 7-1 Generated CSECTs (Page 1 of 2)

Generated For:	CSECT Name	Contents
Each Input Screen	@@IV00mm	Code to execute input validation and data base matching for screen mm.
	@@IP00mm	Code to execute in-storage updates (screen field actions and inserts/deletes) for screen mm.
	@@EH00mm	Error handling code for screen mm.
Each Output Screen	@@OP00mm	Code to build output screen mm.
Individual Request or Procedures	@@R1nnnn	Code to execute a type 1 request/procedure.
	@@R2nnnn	Code to execute a type 2 request/procedure.
	@@RMnnnn	Code to execute a type M request/procedure.
	@@ROnnnn	Code to execute a type O request/procedure.
	@@RSnnnn	Code to execute a type S request/procedure.
Note: The nnnn represents an internally-generated request or procedure number.		

Figure 7-1 Generated CSECTs (Page 2 of 2)

Processing Common to Generated Applications

This section describes processing characteristics common to all generated applications, regardless of the execution environment.

Initial Entry to a Transaction

The initial screen of a VISION:Transact transaction can contain data values or just literals depending on how the terminal user initiates the VISION:Transact transaction. A terminal user can enter the VISION:Transact transaction code with perhaps a file key or range of file keys and the initial screen output by the VISION:Transact transaction can contain data values from the requested record(s).

The following format can be used for initial entry to a VISION:Transact transaction along with pressing the Enter key:

TRANCODE [scrnname] [/ or /DATA]

Where:

TRANCODE	Is the name of the VISION:Transact transaction to be invoked. TRANCODE is 1-8 characters for IMS TM and 1-4 characters for CICS TS. If only the tran code is entered, then the screen format (literals only) for the default initial screen (first input screen named in screen control) is displayed.
scrnname	Is the name (1-8 characters) of a VISION:Transact screen. If only the tran code and screen name are entered (no slash or data), then the screen format (literals only) for the requested screen is displayed. Screen name is optional but when entered must follow the tran code. Tran code and screen name must be separated by a blank.
/ or /DATA	An optional slash (/) can follow the tran code or screen name. When a slash is entered, the default initial screen (the first input screen named in screen control) or the screen name specified is processed by the VISION:Transact transaction as an input screen and screen control is executed to determine the screen to display. With the slash, a complete VISION:Transact interaction is executed with either the default initial screen or the requested screen processed as input. This means that all input processing for the screen is performed (input requests, screen segment, and field actions, etc.), followed by execution of screen control, and the building of the selected output screen displays. Without the slash, a VISION:Transact interaction is not executed; only the screen format (literals only) for the screen is displayed.

Data values can optionally follow the slash. Each data field value entered must be separated by a slash. The data field values entered are moved into the default initial screen or the screen specified before it is processed as input. Data field values must be entered in the same sequence as the input screen fields occur in the screen.

For an example of the initial entry format, suppose you want to execute transaction M501 and provide the key value of the record to be displayed on the subsequent screen. The following would be entered:

M501/11111

This causes the data value of 11111 to be moved to the first input screen field of the default initial screen before it is processed as an input screen. The input screen processing causes data base retrieval (e.g., a match action was specified), screen control logic selects the output screen to be displayed, and the output screen displays data fields from the requested record.

Note: For IMS TM, the /FORMAT command can also be used to request IMS to output the formatted screen for the requested MOD.

Redisplaying the Input Screen

The circumstances under which the generated application redisplays the input screen include:

- An error in screen field validation, based on your coding of S0/Sn statements.
- Inability to retrieve a requested data base segment.
- Inability to update a field in the manner requested.
- An error during the physical update of the data base.
- Execution of an MS operator (requests) or MSG commands (procedures).

Before the screen is redisplayed, the modified data tag (MDT) is set for every screen field which was input to the application; that is, every screen field whose I/O specification is I or B. This is done to prevent the terminal user having to re-key the fields to input them to the application after correcting errors.

Circumstances Under Which a Record or Segment is Re-read

On a second reference to a field in a record, IMS data base segment, or relational data base table which initiates a new loop on the segment, the application generates a request to re-read the record or segment when one of the following conditions exists:

- Data base segment retrieval has been stopped before all requested segments have been retrieved. This can be caused, for instance, by the execution of a NEXT REQUEST branch.
- A field in a screen synchronized data base segment is referenced during output processing and that segment or its parent had been released from the intermediate structure by the SG SELECT ALL operator in screen control.
- The field is in a data base being browsed (browse DSDB) and the browse is not currently active (see the *Advantage VISION:Transact Reference Guide*).
- A file defined as both an SSDB and DSDB is read if referenced in a type 1 request, read based on segment actions in the screen definition, and re-read during output processing.

Whether the request to re-read the record or segment actually results in the execution of a physical I/O operation depends on the facilities of the access method used to read the data base DB2, VSAM, or DL/I.

The input screen can specify a single, unique key value for a segment more than once. There is no restriction against duplicate key processing as long as the processing is consistent with the segment action being attempted. For instance, a single occurrence of a data base segment can be updated twice during the processing of an input screen, but a data base segment occurrence cannot be inserted twice or deleted twice. The generated application retrieves the occurrence subject to this duplicate qualification only once, and updates the single copy of the segment occurrence as many times as requested by the terminal user.

IMS TM PCB Selection

By one means or another (depending on whether the execution environment is IMS TM or CICS TS), the application receives a list of PCBs from the host system. This list is ordered in the same sequence as the PCBs were coded in the PSBGEN (except that for IMS TM an I/O PCB has automatically been inserted at the top by IMS).

When a VISION:Transact application receives control, it attempts to assign one PCB for every data base referenced on an RF statement.

No PCB is shared. This means that the application does not select the same PCB for two or more different data bases.

The following sections detail the two methods of data base PCB selection available to the VISION:Transact generated application. You can be informed of the PCB assigned to each data base by VISION:Transact if your application is generated with execution tracing and application flow tracing is requested at level 2.

Standard Data Base PCB Selection

When the standard method of PCB selection is used, PCBs are selected by DBD name. No attempt is made to find the PCB with the best fit of processing options; any PCB with processing options that are "permissive enough" is eligible to be assigned to a given data base.

"Permissive enough" means that the processing which this application will do on this data base is allowed by the processing options coded in the PCB. For instance, if the PCB processing options include a code of A, that is permissive enough for any processing that a VISION:Transact application might attempt.

If, on the other hand, the VISION:Transact application is not going to update the data base (no actions I, D, U, or V in any screen, or update was not specified for a screen synchronized data base, or the data base is a dynamically synchronized data base), then any processing options whatsoever are permissive enough. Otherwise, if replaces will occur (action U or V), then the PCB processing options must include the code R; if inserts will occur (action I or V), then the processing options must include the code I; and if deletes will occur (action D), then the processing options must include code D.

The algorithm, in procedural form:

1. Perform Steps 2 through 8, considering first the lowest file number and then each additional data base in turn.
2. If this data base is not used in this application, go to Step 8.
3. Look at the first PCB in the parameter list.
4. If this PCB does not have the right DBD name for this data base, or this PCB has already been assigned, go to Step 6.
5. If this PCB has permissive enough processing options, assign it to this data base and mark the PCB assigned.
6. If we did not assign this PCB to this data base and there are more PCBs in the parameter list, move to the next PCB in the list and return to Step 4.
7. If we did not assign a PCB to this data base, issue a diagnostic message indicating no usable PCB found and terminate the application.
8. If we have not reached FILE9 yet, move to the next data base to be considered and return to Step 2.

Positional Data Base PCB Selection

The RC statement parameter Positional PCB? [37] is used to indicate that positional PCB selection is to be used by the generated application. A Y in this column indicates that the selection of the PCBs are to be based on their position in the PSB.

1. The first data base PCB is for the lowest file number.
2. The second data base PCB is for the second lowest file number, etc.

When this method of PCB selection is chosen, no validation of DBD name nor processing option is performed. It is the responsibility of the application developer to provide the DB PCBs in the proper order. The PCBs for the files are selected in ascending file number order regardless of the physical order of the RF statements.

IMS DB Processing

A DL/I segment search argument (SSA) is constructed for every destination field for which a segment action is specified in a screen processing definition. For this reason, every field so specified must be defined as a key or search field to VISION:Transact. See the *Advantage VISION:Transact Reference Guide* for more details on the definition of IMS data bases.

The basic CALL strategy for a generated application is described in the following paragraphs. You can view the DL/I data base calls being issued if your application is generated with execution tracing and SSDB or DSDB I/O call tracing enabled.

Access to screen synchronized data bases (SSDB):

- The Q command code is always used in calls to retrieve SSDB segments. The MAXQ parameter in the recommended PSB for the application reflects this usage.
- Path calls (command code D) are used whenever possible for read and write operations (segment retrieval, replacement, and insertion).
- The DL/I PATH CALL USAGE entry [54] on the RF statement allows path insert calls to be suppressed for a file; path calls will be used for segment retrieval and replacement only. Path insert calls must be suppressed when one of the segments in a path is an LCHILD.
- Path calls are not used to retrieve SSDB segments under two circumstances:
 1. If an SSDB field is used in screen processing definition field validation - that validation being done before the customary retrieval of SSDB segments.
 2. If an SSDB field is used as the default value for a screen field - defaults also being applied before the customary retrieval of SSDB segments.
- An SSDB segment that has a segment action applied to it is retrieved by key (DL/I Get Unique call). A segment that is not referenced with a segment action but is referenced in a request is retrieved with a Get Next call; to retrieve the first occurrence of the lower level segment an F command code is used.
- A single SSDB segment occurrence is never retrieved more than once. If a single occurrence is updated more than once on a screen display, one occurrence is brought into memory and is updated twice.
- By default, if any attempt to update an SSDB segment fails, all previous updates are backed out via the DL/I ROLB call. The input screen is then redisplayed with an appropriate error message. Use of the ROLB call requires at least DL/I Release 1.5. The ROLLBACK option of the SYNCPOINT command under CICS TS, which backs out all data base updates since the last update, performs this backout function in that environment. Note that this data base backout can be suppressed on an individual application basis (see the SYSTEM ROLLBACK CONTROL entry [33] on the RC statement and the PR statement MS operator for requests or the MSG command for procedures).

Access to dynamically synchronized data bases (DSDB):

- Neither the Q command code nor path calls are used; however, a D command code is used in the SSA for the root data base segment.
- Root segments are retrieved with a GU call that is qualified only on the key of the root segment. For browse synchronized DSDBs, two segment search arguments (SSAs) are used with GE and LE operators in the Get Unique call.
- Get Next calls are used to retrieve lower level segments. To retrieve the first occurrence of lower level segment the F command code is used.
- Only one record per DSDB is in memory at a time.
- Reference to a field in a DSDB under browse synchronization always causes the browse operation to begin, unless the browse is already in progress.
- Reference to a field in a lower level segment of a DSDB does not cause a re-read of the segment type, if a GE status code was returned the last time the segment type was accessed and the value of the synchronizing field has not changed since the last reference.

Relational Data Base (DB2) Processing

SQL statements are generated to perform the specific table actions required by the application. These statements are output to the M5PREP data set by the generator and pre-compiled during the application installation process. Commits are synchronized through the normal IMS TM and CICS syncpoint processing. General guidelines for DB2 performance considerations also apply to VISION:Transact applications using relational data bases. Among these should be consideration for creating indexes for columns used in search conditions. Refer to the *Advantage VISION:Transact Reference Guide* for additional information.

You can view the SQL data base calls being issued if your application is generated with execution tracing and SSDB or DSDB I/O call tracing is enabled.

IMS TM PROCESSING

The storage requirements for the region in which the generated application executes can be estimated from the sum of the following:

- The size of the link edited application load module.
- The size of the VISION:Transact IMS TM executor routines only if they are dynamically loaded during execution (M5PARM XLOAD=Y) and not link edited with the generated application.
- Additional main storage required by the application for initial screen data areas (screen buffers) and internal work areas. The exact amount of storage is application dependent and is determined by VISION:Transact.

- The file and work area buffers required for any execution of the application. The amount of storage required for the application's buffers may vary from one interaction to the next, depending on the files and work areas used during any interaction. This amount includes temporary fields and constants, working storage, screen and dynamically synchronized data base segments, as well as pseudo file segments referenced by the input or output screen, or by screen requests. VISION:Transact determines a minimum buffer size and obtains extra storage as needed in 2K increments.
- The size of any called modules that are loaded dynamically during application execution (that is, as opposed to called modules that are included when the application is link edited, which are loaded with the application).

For a conversational program the size required for the SPA is the size of the working storage and pseudo files specified as SPA-resident (storage type of S) plus 24 bytes, which are used by the generated application. The required value is printed in the APPGEN listing. [Figure 7-2](#) shows the SPA format used by VISION:Transact.

Positions	Descriptions
1-14	Standard IMS control fields: 2 bytes: length 4 bytes: reserved for IMS 8 bytes: name of receiving transaction
15	Flag byte required by VISION:Transact which contains one of the following values: - X'00' Indicates the first interaction of the conversation. IMS TM has removed the transaction code and a blank from the input message and placed it in the SPA. Data values in the SPA have not been initialized. - X'80' Indicates a subsequent interaction of the conversation. The SPA has been previously written out with the contents of the saved working storage and/or pseudo file fields within the application. X'A0' Indicates a subsequent interaction of the conversation. The SPA has been previously written out by a VISION:Transact transaction initiated from entering transaction code [screenname]. The contents of the SPA have not been initialized.
16-24	Reserved by VISION:Transact (not currently used).
25	Start of type S WORKnn areas followed by pseudo files in sequence by their file type.

Figure 7-2 IMS TM SPA Format

Redisplay of the input screen by the application:

- Is always through an insert using the I/O PCB.
- By default, causes a DL/I rollback (ROLB). When the RECYCLE option of screen control is used, there may be several update cycles in an interaction, all of which by default are backed out.

Note: The DL/I rollback can be suppressed on an application basis by the System Rollback Control entry [33] on the RC statement and the PR statement MS operator for requests or the MSG command for procedures.

CALL, DLI, and PLI Statements (IMS™)

You can use VISION:Transact to call subroutines written in other languages such as Assembler, COBOL, or PL/I. You can also invoke DL/I directly.

VISION:Transact application programs run as IBM Language Environment (LE) compliant programs within an active LE environment (enclave). All LE services compatible with the host environment (CISC TS or IMS™) are available within called subroutines.

Use the CALL or PLI statement (requests) or CALL MODULE command (procedures) to invoke a user written subroutine. PLI is functionally identical to CALL except that it follows the PL/I convention for passing character arguments.

Use the DLI statement (requests) or CALL DLI command (procedures) to execute a DL/I function. DLI is functionally equivalent to “CALL ASMTDLI” (requests) or CALL MODULE ASMTDLI (procedures) but does not require a LOAD for ASMTDLI.

Linkage to user subroutines can be either static or dynamic. With static linkage, the subroutine is included in the application load module. There are two ways to specify static linkage. First, you can specify a value of Y in the LKEDIMS keyword in M5PARM to indicate that all programs are to be link edited with the application. This value can be overridden by the LKEDCALL Programs entry [32] on the RC statement during application generation. (See the *Advantage VISION:Transact Installation Guide* for information on M5PARM.) Secondly, you can specify static linkage by using the LKEDCALL RP statement option. You may need to provide the appropriate linkage editor INCLUDE control statements.

Note: If your CSECT name is different than your load module name, use the CSECT name on the CALL statement and name the member name on the INCLUDE. If the CSECT name and member name are the same, you can remove the NCAL parameter and supply your loadlib as a concatenation to SYSLIB.

PL/I programs may not be statically linked. If a program which is to be statically link edited with the VISION:Transact application is unresolved (not available) at link edit time, VISION:Transact will dynamically load the program during application execution.

With dynamic linkage, the subroutine is a separate load module. The application automatically loads it into virtual storage the first time it is called. The subroutine module then remains in storage until the VISION:Transact application deletes it. The application automatically deletes any loaded modules just before terminating. Dynamic linkage is indicated when a value of N is specified for the LKEDIMS keyword in M5PARM or for the LKEDCALL Programs entry [32] on the RC statement.

Calls do not affect the validity or invalidity of a field specified as a parameter. A field which is valid prior to a call is assumed valid by VISION:Transact following the call.

Calling Assembly Language Subroutines (IMS™)

When an IMS™ application executes a call, a parameter list is constructed with the address of each parameter (if any are specified) appearing in the address list in the order specified in the CALL statement. Following standard call conventions, the last address in the list has a hexadecimal 80 in the high order byte to indicate the end of the list. Also adhering to standard programming practices, you cannot assume that the address of a parameter remains the same from call to call. When a call is generated, VISION:Transact observes the following call conventions:

- The entry point address of the called routine is placed in register 15.
- The return point to the VISION:Transact application is placed in register 14.
- The address of the save area of 18 full words is placed in register 13.
- The address of the parameter list is placed in register 1.
- Control is passed to the named module (or CSECT, when you use static linkage) and upon return, the contents of register 15 are placed in the VISION:Transact return code flag.

The following points should also be observed when coding assembler language subroutines:

- Your subroutine is responsible for saving and restoring the contents of (at a minimum) registers 11 through 14. You can accomplish this by using the IBM SAVE and RETURN macros.
- If the called routine does not handle interrupts and an interrupt occurs, the VISION:Transact application terminates with a User 60 ABEND. You can establish your own interrupt handling by use of the ESPIE macro; however, you must restore the VISION:Transact PICA before returning to the VISION:Transact application. [Figure 7-3](#) shows an example of restoring the VISION:Transact PICA.
- User routines are assumed to be reusable, and may be executed several times without being reloaded. Non-reusable routines require an interface program to ensure a fresh copy for each execution.

...	Provide exit routine for fixed point overflow
ESPIE SET, FIXUP, (8)	
ST 1, HOLD	Save address returned in register 1
...	
L 5, HOLD	Reload returned address
ESPIE RESET, (5)	Use execute form and old PICA address
HOLD DC F`0'	

Figure 7-3 Restoring the PICA

Calling PL/I Subroutines (IMS TM)

Applications generated for the IMS TM environment can issue calls to PL/I subroutines by including the PLI statement in requests or the CALL MODULE command with the PLI keyword in procedures. Parameters defined to VISION:Transact as type C fields can be passed via indirect addresses to conform to PL/I conventions. The CALL statement (requests) can also be used or the PLI keyword (procedures) can be omitted if character arguments are not being passed.

The following conditions must be met for successful invocation of PL/I subroutines by VISION:Transact generated applications:

1. OPTIONS(MAIN) must be coded in the PL/I PROCEDURE statement.
2. The PL/I environment must be set up when the subroutine is called. This means that when the PL/I subroutine is link edited, it must have an entry point of PLICALLA (such as, ENTRY PLICALLA).
3. No variable (VAR) character strings may be passed to or from the subroutine, but CHAR(*) is usable (and recommended) for parameters that are passed between the two programs.
4. The PL/I LINKLIB must be available to the IMS TM system.
5. Static linkage (LKEDCALL RP option) is not supported for PL/I subroutines.

Calling COBOL Subroutines (IMS TM)

You can use VISION:Transact to call subroutines written in COBOL, subject to the following restrictions:

1. The COBOL compiler options RESIDENT, DYNAM, and ENDJOB cannot be used.
2. If the COBOL subroutine calls other subroutines, only the "CALL literal" form can be used.
3. Certain COBOL statements which cause dynamic storage to be acquired should not be used because the storage is not released. These statements are: INSPECT, UNSTRING, and SIGN IS SEPARATE.

Use the GOBACK statement to return control to VISION:Transact.

Passing Control to Another IMS TM Transaction

A VISION:Transact transaction can pass control to another IMS TM transaction (VISION:Transact or non-VISION:Transact) through the use of the TRANCODE flag on the screen or through the use of Operand C in screen control.

If the TRANCODE flag is used on the screen, its content determines the IMS TM transaction that is to be invoked when the screen is input from the terminal. A VISION:Transact application can move a new transaction code into the TRANCODE flag prior to outputting a screen to the terminal (e.g., in a type O request for the output screen). When the screen output by the VISION:Transact transaction is input from the terminal, the new transaction (the transaction named in the TRANCODE flag) will be invoked. This method of passing control to another transaction is often referred to as a deferred program message switch. Note that the input transaction code on the SD statement can be used instead of the TRANCODE flag to indicate the transaction that is to be invoked when the screen is input from the terminal, or in addition to the TRANCODE flag to supply a default input transaction code when the TRANCODE screen field is left blank.

Operand C in screen control can specify the name of a transaction to which control is to be passed along with the output screen. This is referred to as an immediate program message switch. The VISION:Transact transaction sends the output screen as a message to the transaction named in Operand C of screen control; an ISRT call to a modifiable alternate TP PCB is issued by VISION:Transact. When a program- to-program message switch results in an IMS security violation, VISION:Transact redisplay the input screen with a message.

Certain rules exist for program message switching depending on whether the transaction passing control or receiving control is non-conversational or conversational. A conversational transaction uses the IMS TM Scratch Pad Area (SPA). A VISION:Transact transaction is conversational if it uses any saved working storage areas or saved pseudo files (Storage Type entry of S on the RF statement) as they are saved between interactions in the SPA.

The following rules exist for deferred and immediate program message switching:

- A non-conversational program can perform a deferred or immediate message switch to another non-conversational program. When an immediate message switch is used (Operand C of screen control), the XIND parameter of the OC operator in screen control should be used to suppress the VISION:Transact missing, invalid, overflow, and null field indicators in the output screen (output message). The XIND parameter is required when Operand C specifies a VISION:Transact transaction.
- A non-conversational program can only do a deferred switch to a conversational program. IMS TM does not allow a non-conversational program to do an immediate message switch to a conversational program.

- A conversational program can perform a deferred or immediate message switch to another conversational program. When an immediate message switch is used (Operand C of screen control), the CONV parameter of the OC operator in screen control must be specified. The CONV parameter causes VISION:Transact to ISRT the SPA and the message in the proper sequence for the message switch. In addition, the XIND parameter of the OC operator should be used to suppress VISION:Transact missing invalid, overflow, and null field indicators in the output message. The XIND parameter is required when Operand C specifies a VISION:Transact transaction. Note that if output to alternate terminals is required and the conversation is to be passed to another program (message switch), screen control must code all output to alternate terminals first followed by the message switch; the message switch must be specified last.
- A conversational program should use a deferred message switch when control is to be passed to a non-conversational program. This ensures that IMS properly discards the SPA. When a deferred message switch is used in VISION:Transact, blanks should be moved to the TRANCODE flag prior to outputting the screen to the terminal; this must be done in the type O request for the output screen. This causes IMS to discard the SPA when the output message is sent. If the screen being output to the terminal by VISION:Transact has an input transaction code specified on the SD statement, then the screen is input to that transaction regardless of the contents of the TRANCODE flag. However, if the screen uses the TRANCODE flag instead of the input transaction code on the SD statement, then the contents of the TRANCODE flag on the screen determines the transaction to be invoked when the screen is input. Since the TRANCODE flag needs to be set to blanks to discard the SPA, it will contain blanks when the screen is output by VISION:Transact. Thus, when the TRANCODE flag is used on the screen and a deferred switch from a conversational to a non-conversational program is desired, the user must key in the desired transaction code before entering the screen.

Note: When VISION:Transact passes control to a non-VISION:Transact transaction, the format of the SPA and message output by VISION:Transact is shown in [Figure 7-2](#) and [Figure 7-4](#), respectively. When a non-VISION:Transact transaction passes control to a VISION:Transact transaction, the SPA and message must be output according to the format VISION:Transact expects (see [Figure 7-2](#) and [Figure 7-4](#)).

MFS Features Supported

The MFS control statements generated by VISION:Transact support the following features:

- Multiple device format definitions per screen. A DIF/DOF is generated for each device type specified in the M5PARM terminal device table group (see the *Advantage VISION:Transact Installation Guide*).
- 3270 display devices and printers and SCS printer devices.

- The implementation of a dynamic transaction code through the use of the TRANCODE flag on the screen. The transaction code is inserted in front of the message; this is done by moving the DFLD pointing to the TRANCODE flag to the first MFLD.
- PF keys, using a generated PFK= control statement to insert a literal into the message with the value of the PF key pressed.
- Additional device parameters for the DEV statement as specified in the M5PARM device table.
- Operator Logical Paging of an output screen display. The MFS paging command must be the first field in the input message. To accomplish this, the terminal operator enters the installation's MFS paging command into a field defined in the root segment of the screen with destination of F,TRANCODE.
- Dynamic modification of attributes.

Message Format and Attributes

[Figure 7-4](#) illustrates the characteristics of the message format produced by VISION:Transact. Note that during online execution, IMS TM inserts the 4-byte length field (LLZZ) in front of each message.

The following is true of the message format produced with VISION:Transact and the control block descriptors processed by a VISION:Transact generated application:

- The input and output messages are the same format for the redisplay of the input screen during error processing.
- A two-byte attribute precedes each field in the message.

Positions	Descriptions
1-9	Transaction Code: eight characters followed by a blank.
10-17	Screen Name.
18-22	ATTNID flag.
23-101	Message.
102-...	Fields preceded by two attributes.

Figure 7-4 Message Format

- Modify data tag (MDT) attributes are set by the application. The MDT is set at output time for screen fields with an I/O specification of B - both input and output. The MDT is set upon the redisplay of the input screen for input screen fields with an I/O specification of I or B.
- The application program always expects a single-segmented input message.

- The application program produces a single-segmented output message, except when A is specified for output control in the screen processing definition or OC OUTLIM is specified in screen control, which causes a multiple-segmented message to be produced with one segment per screen display.
- If the XIND screen control option is not specified, hex "3F" is placed in a field to cause the device field not to be erased and hex "403F" is placed in a field to cause the device field to be erased.

CICS TS PROCESSING

The storage requirements for the generated application can be estimated from the sum of the following:

- The size of the link edited application load module.
- The size of the VISION:Transact CICS TS executor modules that are dynamically linked during application execution.
- The size of all working storage areas which are in the CICS TWA, as listed by VISION:Transact in the generated PCT entry, multiplied by the number of concurrent users.
- Additional main storage required by the application for initial screen data (screen buffers) and internal work areas, multiplied by the number of concurrent users. The exact amount of storage to be multiplied is application dependent and is determined by VISION:Transact.
- The file and work area buffers, also multiplied by the number of users, required for any execution of the application. The amount of storage required for the application's buffers can vary from one interaction to the next, depending on the files and work areas used during any interaction. This amount includes temporary fields and constants, screen and dynamically synchronized data base segments, as well as pseudo file segments referenced by the input or output screen, or by screen requests. VISION:Transact determines a minimum buffer size and obtains additional storage as needed in 2K increments.
- The size of any called modules, which are linked dynamically during application execution.

The generated application gains exclusive control of the VSAM control interval for a data base record only during update processing. Processing of VSAM files is discussed in the *Advantage VISION:Transact Reference Guide*.

The generated application uses CICS TS temporary storage for at least two purposes:

- To save the follow-on TRAN code and screen name to enable pseudo conversational processing. This record (the screen record) is written to main temporary storage.
- To store program information between interactions. (Storage type S for working storage or pseudo files.) The saved working storage record and pseudo file records are written to auxiliary temporary storage.

[Figure 7-5](#) shows the contents of the temporary storage records used for both of these purposes.

Note: Working storage (Storage Type S) specified for an application is stored in a single record. Pseudo file(s) (Storage Type S) on the other hand, are stored as separate records. For example, if your application has WORK01 through WORK04, those four files are stored in the same record. In addition, if that same application has three pseudo files specified, each of those pseudo files is stored in a separate record.

Note: A VISION:Transact application generated with execution tracing uses CICS TS temporary storage to save the contents of the trace flags between interactions. Applications generated with tracing may also use CICS TS temporary storage and/or transient data as a destination for trace output. For more information, see [Execution Tracing of CICS TS Applications on page 6-7](#).

SCREEN or COMMAREA SECTION RECORD

Queue name if terminal program - SCRNTerm

SCRN	Literal SCRN.
term	Four-byte terminal ID.

Queue name if COMMAREA program - COMMconversation

COMM	Literal COMM.
conversation	Twelve-byte conversation ID from COMMAREA. (See Transaction Control Fields in a COMMAREA in Chapter 5 of <i>Advantage VISION:Transact Reference Guide</i> .)

This queue only contains a single word.

Record length - 12

Record format - transcrnname

tran	Transaction ID (four characters).
scrnname	Screen or COMMAREA section name (eight characters).

WORKING STORAGE RECORDS AND PSEUDO FILE RECORDS

Queue name if terminal program - SWK5term

SWK5	Literal SWK5.
term	Four-byte terminal ID.

Queue name if COMMAREA program - SWKCconversation

SWKC	Literal SWKC.
conversation	Twelve-byte conversation ID from COMMAREA. (See Transaction Control Fields in a COMMAREA in Chapter 5 of <i>Advantage VISION:Transact Reference Guide</i> .)

The queue can contain multiple records; a single record is used for all saved working storage areas (Storage Type S). The saved working storage record is written first if present, followed by each saved pseudo file record. Saved pseudo file records are written in sequence by their file type.

Record length -	Must equal the total length of either all saved working storage areas or an individual saved pseudo file record, plus 4 bytes.
-----------------	--

Record format - zzzzdata

zzzz	Reserved (four bytes) for saved working storage only; four bytes are not reserved for a saved pseudo file record.
data	All saved working storage definitions or an individual pseudo file definition.

Figure 7-5 CICS TS Temporary Storage Contents Used by a Generated Application

CALL and DLI Statements (CICS TS)

You can call subroutines written in other languages such as Assembler or COBOL from your VISION:Transact application. You can also invoke DL/I directly.

Use the CALL statement (requests) or CALL MODULE command (procedures) to invoke a user written subroutine. Use the DLI statement (requests) or CALL DLI command (procedures) to execute a DL/I function.

Linkage to user Assembler language subroutines can be either static or dynamic. With static linkage, the subroutine is included in the application load module. There are two ways to specify static linkage. First, you can specify a value of Y in the LKEDCIS keyword in M5PARM to indicate that all programs are to be link edited with the application. This value can be overridden by the LKEDCALL Programs entry [32] on the RC statement. (See the *Advantage VISION:Transact Installation Guide* for information on M5PARM.) Secondly, you can specify static linkage by using the LKEDCALL RP statement option. You may need to provide the appropriate linkage editor INCLUDE control statements.

Note: If CSECT and member name of the load module is the same, you can remove the NCAL parameter and supply your LOADLIB as a concatenation to SYSLIB. If your CSECT name differs from your load module name, use the CSECT name on the CALL statement and name the member name on an INCLUDE statement in the link edit.

Statically linked subroutines must be written in Assembly language and cannot contain CICS commands or macros. For a statically linked subroutine, parameters are passed on the CALL statement. The subroutine must observe standard IBM linkage conventions. On entry to the subroutine:

- Register 15 contains the subroutine entry point.
- Register 14 contains the VISION:Transact application return address.
- Register 13 contains the address of a standard register save area.
- Register 1 contains the address of a parameter list.

With dynamic linkage, the subroutine is a separate load module with its own entry in the CICS CSD. This is the recommended method to use under CICS. The subroutine can be written in any language supported by CICS and has access to all CICS functions. To return control to the VISION:Transact generated application, execute a CICS RETURN command. Dynamic linkage is indicated when a value of N is specified for the LKEDCIS keyword in M5PARM or for the LKEDCALL Programs entry [32] on the RC statement.

Data can be passed to a dynamically linked called user subroutine through the CICS COMMAREA or the CICS TWA.

If data is to be passed through the CICS COMMAREA, the VISION:Transact CALL statement must specify two parameters. The first parameter names the field that defines the first byte of the COMMAREA. The second parameter must define a two-byte binary field that contains the length of the COMMAREA to be passed to the called program.

If data is to be passed through the CICS TWA, then no parameters are specified on the VISION:Transact CALL statement. In this case, data is placed in VISION:Transact working storage fields as they reside in the CICS TWA.

The VISION:Transact generated application's working storage (WORKnn files on the RF statement) starts at the beginning of the TWA. Multiple working storage definitions are laid out in the TWA in ascending sequence by their “nn” specifications in the RF statements.

Passing Control to Another CICS TS Transaction

A VISION:Transact transaction can pass control to another CICS TS transaction (VISION:Transact or non-VISION:Transact) through the use of the TRANCODE flag or through the use of the START parameter of the OC operator in screen control.

A VISION:Transact application can move a new transaction code into the TRANCODE flag prior to outputting a screen to the terminal (e.g., in a type O request for the output screen). When the screen output by the VISION:Transact transaction is input from the terminal, the new transaction (the transaction that was named in the TRANCODE flag) is invoked. The value in the TRANCODE flag is inserted in the TRANSID parameter of the CICS RETURN command during output processing by the VISION:Transact transaction, which causes the new transaction to be invoked on input. This is a deferred method for passing control to another transaction as the new transaction is not invoked until the screen output by the VISION:Transact transaction is input from the terminal. Note that the input transaction code on the SD statement can be used to indicate the transaction that is to be invoked when the screen is input from the terminal.

The START parameter of the OC operator in screen control can be used to pass control and the output screen to another CICS TS command level transaction. The transaction code of the transaction to be started is moved into the TRANCODE flag prior to issuing the OC START in screen control. Once the processing of the VISION:Transact transaction completes, the new transaction (VISION:Transact or non-VISION:Transact) is started via CICS interval control. VISION:Transact passes the output screen and any saved working storage and/or saved pseudo files (Storage Type entry of S on the RF statement) to the started transaction. If the started transaction is a VISION:Transact transaction, the passed output screen is processed as an input screen just as if it had been input from the terminal.

The OC operator with the START parameter is specified in screen control after an OS operator. The OS operator names the output screen that is to be passed to the started transaction. When OC START is specified, VISION:Transact issues a CICS START command for the transaction named in the TRANCODE flag. A transaction (task) that is started by VISION:Transact is always associated with a terminal.

Note: The OS operator that precedes the OC START operation can use Operand C to specify the terminal ID to be used for the started transaction. When Operand C of the OS operator is left blank, the transaction will be started for the originating terminal.

VISION:Transact passes separate data records to the started transaction for the output screen, saved working storage, and each saved pseudo file.

VISION:Transact issues multiple CICS START commands (one for each data record to be passed) with the FROM option; each START command specifies the same transaction code and terminal ID. The started transaction (command level) uses the CICS RETRIEVE command to access the data records passed by VISION:Transact. A separate RETRIEVE command with the INTO option must be issued by the started transaction to obtain each data record passed. A CICS transaction can interrogate the STARTCODE field (available through the ASSIGN command) to determine how it was initiated. If the STARTCODE field is equal to SD (transaction initiated by a START command with data), the appropriate RETRIEVE command(s) can be issued. When the started transaction is a VISION:Transact transaction, the appropriate RETRIEVE commands are automatically issued.

The first data record that is always passed by VISION:Transact to the started transaction is the BMS map for the output screen. This data record has the following format:

Bytes:

1-4	Contains the four-byte transaction code of the transaction that issued the START command.
5-12	Contains the eight-byte output screen name.
13-19	Reserved seven bytes.
20-24	Current value of the ATTNID flag.
25-End	Contains the mapped output screen data starting in byte 25.

If the VISION:Transact application uses any saved working storage areas, they will be passed next as a single data record. The format of this record is determined by the file definitions for "all" saved working storage areas; all saved WORKnn definitions are concatenated in sequence by their File Type entry (such as, WORK01, WORK02, WORK03. . .) and passed as a single data record.

Subsequent data records are passed for each saved pseudo file used in the VISION:Transact application. Saved pseudo file records are passed in sequence by their File Type entry. The format of each record is determined by its file definition.

Note that the data record containing the BMS map for the output screen is always passed to the started transaction. Additional data records (one for the saved working storage areas and one for each saved pseudo file) are only passed when present in the VISION:Transact application.

A non-VISION:Transact transaction can pass control to a VISION:Transact transaction by using the CICS START command in the same manner as is done by VISION:Transact when starting a transaction. The START command should associate the transaction with a terminal (TERMIN option). The first START command must pass the BMS map data record for an output screen using the FROM option. The format for this data record is shown above. Subsequent START commands can be used to pass additional data records using the FROM option. Each START command passes one record. The additional data records must match the VISION:Transact saved working storage areas or an individual saved pseudo file record.

BMS Features Supported

VISION:Transact generates one BMS mapset for all screens included in a single definition run or for all screens in an APPGEN run, whose mapset name is determined by the first screen identifier in the run. It is strongly recommended that you define only one screen per definition run for CICS TS-targeted transactions to allow yourself the maximum flexibility and maintainability for your BMS maps and mapsets.

The generated BMS specifies TERM=3270 on the DFHMSD macro which supports all 3270 models. The screen height and width from the M5PARM device table is used for the map SIZE parameter on the DFHMDI macro. See the *Advantage VISION:Transact Installation Guide* for information on the M5PARM device table.

BMS routing is used for output directed to an alternate terminal. BMS routing uses CICS temporary storage. The temporary storage requirements in your CICS TS environment can increase if your application sends information to alternate destinations via screen control specifications, which invokes BMS routing facilities.

VISION:Transact supports BMS paging of an output screen display when that screen is defined with A specified for Output Control on the SD statement or the OC OUTLIM parameter is used in screen control. The BMS paging command must be established as the first input field on the screen. To accomplish paging, the terminal operator enters the installation's BMS paging command into the first input field on the screen. BMS paging also uses CICS temporary storage.

Processing Clear and PA Keys

VISION:Transact allows you to control the handling of the Clear and/or PA keys on an installation and individual application program basis. The CICSKEY entry in M5PARM provides the installation default, while the CICSKEY Handling entry [34] on the RC statement provides the application override. Refer to the *Advantage VISION:Transact Installation Guide* for information and specifications for the CICSKEY entry in M5PARM. Refer to the *Advantage VISION:Transact Specifications Guide* for information on the CICSKEY Handling entry on the RC statement.

The following items must be noted concerning the use of Clear and PA keys:

- Since no input screen data is passed to an application program when the Clear or PA keys are pressed, the resulting VISION:Transact input screen contains blanks in all screen fields in screen segment 1. Repeating screen segments are missing unless null screen segment processing is requested on the screen's SD statement.
- A type 1 request or procedure must check for these keys (flag ATTNID equal to CLEAR, PA001, PA002, or PA003) and take appropriate action. It should either:
 1. Issue the MS operator (requests) or MSG command (procedures) to invoke buffer reshow. When the input screen is redisplayed, only literal data, modified input screen fields, and the messages from the MS operator or MSG command are sent to the terminal; the original contents of input data fields are lost when the Clear or PA keys are pressed. Only modified input screen fields are sent to the terminal. These are input screen fields designated as display modify in the screen definition (Display Any Modification entry of Y on the S0 statement) whose contents are changed by the programmer to non-blank prior to the MS operator or MSG command. Note that output-only fields are erased.
 2. Issue a branch or TRANSFER to SCREEN CONTROL to bypass input processing in the interaction. Input processing should be bypassed since the input screen contains no data. Screen control must include logic to output the desired screen when the Clear key or one of the PA keys is pressed.
- Data to be passed between interactions cannot be passed through the screen in fields designated as both input and output since input screen data is lost. If data needs to be passed between interactions, saved working storage or saved pseudo files must be used.
- Terminal users must have a way to exit a VISION:Transact transaction since the Clear and/or PA keys do not cause application termination. Termination of a VISION:Transact transaction can be accomplished by either:
 1. Naming no output screen for the input screen in screen control.
 2. Outputting an exit type screen and setting the TRANCODE flag to blanks in the exit screen's type O request.

Considerations For The Development Of Applications For Multiple Targets

The preceding subsections contain information regarding the execution of VISION:Transact generated applications under CICS TS or IMS TM. When you develop applications that are targeted for both these environments, some areas of VISION:Transact coding warrant particular attention. [Figure 7-6](#) illustrates some of them.

ISSUE	IMS TM	CICS TS
Target Monitor on RC.	I	C
Load Module Name Override.	PSB name on RC statement.	Program name on RC statement.
Message Processing on RC.	Multiple messages.	Ignored.
PF Keys.	PF keys available for all screens.	If the installation BMS is set up to use PF keys for logical paging, NO PF key is available to any screen with Output Control = A.
Clear and PA Keys.	Not available.	Controlled on an installation basis in M5PARM. Controlled on an application basis on the RC statement.
Empty Fields (in a message switch).	Start X'003F' unless XIND on screen control is used.	X'0000'.
Saved working storage and pseudo file records (Storage Type S).	All reside in a single SPA record.	All reside in a single temporary storage queue; a single record is created for all work areas saved and a separate record is created for each pseudo file saved.
Testing.	BTS.	EDF or CICS.

Figure 7-6 Considerations for Multiple Targets (Page 1 of 3)

ISSUE	IMS TM	CICS TS
External Routines.	Dynamically loaded or statically linked.	Usually dynamically loaded (Assembler programs with no CICS commands or macro may be statically loaded).
File Identification on FD.	DBD name.	DBD name or CICS FCT name.
Segment Counters in Hierarchies.	Not supported.	Not supported for DL/I or SQL. Mandatory for VSAM and ISAM.
Trancode.	Eight characters.	Four characters.
Screen Identifier.	One to five characters are valid as well as seven, but not six. One to five must be unique. Can do screens individually.	One to seven characters. Can do screens individually. To combine several screens in one mapset, must catalog together.
Field Separation.	Greater than to or equal one.	One or greater than or equal to three.
CALL Format.	As documented for CALL, PLI, or DLI.	As documented for CALL or DLI.
Logical Paging with Output Control = A on SD statement or OC OUTLIM on SC statement.	Use screen field whose destination is TRANCODE flag.	First input field,, any destination (beware of erroneous update of the TRANCODE flag if it is used).
Paging Commands.	PA1 can wrap around =, =nn, =+n, =n, =L. No copy routing.	N P/F C/....T/ L + n - n
Positional PCB or RF statements with a File Type of PCBnn.	Includes TP PCBs.	No TP PCBs.
Use of the TRANCODE flag to Choose Next Transaction.	Must be on screen and should be protected (eight characters).	Need not be on screen (four characters).

Figure 7-6 Considerations for Multiple Targets (Page 2 of 3)

ISSUE	IMS TM	CICS TS
Automatically Starting a New Program.	Message switch via screen control (Operand C).	Set TRANCODE flag and use START operator in screen control.
Screen Control with No Output Screen.	Old screen stays.	Screen blanked.
Files Available.	IMS DB, DB2, and Fast Path.	IMS DB, DB2, and VSAM (no Fast Path).
Error Destination.	Set in M5PARM or determined by second TP PCB (first one you supply in PSB).	M5PARM.

Figure 7-6 Considerations for Multiple Targets (Page 3 of 3)

Overview

This chapter shows you how to use the programs that service VISION:Transact libraries. These programs provide the means to initialize, dump (backup), restore, and condense a library, and other specialized services. The chapter first illustrates the use of the M5UTIL program that offers a comprehensive set of library services via simple control cards. Then the chapter illustrates the use of the M5INIT, M5DUMP, M5REST, and M5CON programs. Since these programs are specialized and require no control cards, they offer a quick and handy library service.

The first half of the chapter describes the JCL and data sets required by the various operations supported by the M5UTIL program. The second half describes the JCL and data sets used with the M5INIT, M5DUMP, M5REST, and M5CON programs.

The M5UTIL Program

The M5UTIL program provides a comprehensive set of library service operations. It combines the operations provided by the M5INIT, M5DUMP, M5REST, and M5CON programs, and adds a few of its own.

The M5UTIL program requires and accepts utility control (UC) statements. UC statements and comment (AA) statements are the only allowable input statements for M5UTIL. Via the UC statements, M5UTIL provides its services for your use on VISION:Transact libraries. Each particular service that you require of M5UTIL must be coded as an operation on a UC statement. The M5UTIL program provides the following variety of operations:

INIT	To initialize a VISION:Transact library.
DUMP	To dump (backup) a library.
REST	To restore a library that has been dumped.
COND	To condense a library.
COPY	To copy specified items from one library to another.
MERG	To copy all the items from one library to another.

The COPY and MERG operations are only provided by the M5UTIL program, while the INIT, COND, DUMP, and REST operations are also provided separately by the M5INIT, M5CON, M5DUMP, and M5REST programs, respectively. These programs, which are compatible with M5UTIL, are covered later in this chapter.

The M5UTIL program can process multiple UC statements and each can specify the ddname of a different library as the object of its operation. This means you can perform many services on many libraries in a one job step. However, since many of the library operations (such as, dumps and restores) are resource intensive, be careful when you combine them.

The next few sections cover the JCL, data sets, and UC statements appropriate for several common library services provided by M5UTIL. The UC statement specifications are discussed in detail in the *Advantage VISION:Transact Specifications Guide*.

M5UTIL Initialization

Before defining any data to VISION:Transact you must initialize your VISION:Transact library. You may qualify the UC statement with the INIT operation code and M5UTIL will initialize your library. The library to be initialized may be specified in the DD/DTF name field of the UC statement. If the ddname is not specified, M5LIB will be initialized. [Figure 8-1](#) and [Figure 8-2](#) show the JCL necessary to initialize your M5LIB. Before initializing a VSAM M5LIB it is necessary to execute a DEFINE CLUSTER. See [Defining a VSAM M5LIB on page 8-12](#) for the DEFINE CLUSTER JCL.

```

//jobname JOB . . .
//*
//* JCL USED TO ALLOCATE AND INITIALIZE A BDAM M5LIB
//*
1> //init EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.bdam.M5LIB,
// DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
4> //M5WORK DD DUMMY
//M5INPUT DD *
5> UCINIT
//

```

Figure 8-1 Initialize a BDAM Library

```

//jobname JOB . . .
//*
//* JCL USED TO INITIALIZE A DEFINED VSAM M5LIB
//*
1> //init EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.vsam.M5LIB,DISP=OLD
4> //M5WORK DD DUMMY
//M5INPUT DD *
5> UCINIT
//

```

Figure 8-2 Initialize a VSAM Library

Notes	Explanation
1.	The service program M5UTIL is invoked.
2.	The M5LIST statement defines the location of the system output device for this job.
3.	M5LIB represents the library you are initializing in this run.
4.	An M5WORK file is required for all M5UTIL functions. Although required, it is not used by the INIT operation and can be dummied.
5.	The UC statement must be coded with the operation code INIT to initialize a library.

M5UTIL Dump

The operation code DUMP is used to create an unloaded sequential copy of a library. The library to be dumped may be specified in the DD/DTF name field of the UC statement. If the ddname is not specified, M5LIB is dumped. Refer to the *Advantage VISION:Transact Specifications Guide* for more information on the UC statement. [Figure 8-3](#) and [Figure 8-4](#) show the JCL necessary to dump (backup) a BDAM and VSAM library.

```
//jobname JOB . . .
//*
/* THIS JCL IS USED TO DUMP A BDAM M5LIB
/*
1> //dump EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.bdam.M5LIB,DISP=SHR
4> //M5WORK DD DSN=your.dump.M5WORK,DISP=(NEW,CATLG,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(n,n),RLSE)
//M5INPUT DD *
5> UCDUMP
//
```

Figure 8-3 Dump a BDAM Library

```
//jobname JOB . . .
//*
/* THIS JCL IS USED TO DUMP A VSAM M5LIB
/*
1> //dump EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.vsam.M5LIB,DISP=SHR
4> //M5WORK DD DSN=your.dump.M5WORK,DISP=(NEW,CATLG,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(n,n),RLSE)
//M5INPUT DD *
5> UCDUMP
//
```

Figure 8-4 Dump a VSAM Library

Notes	Explanation
1.	This statement invokes M5UTIL.
2.	The M5LIST statement defines the location of the system output device for this job.
3.	M5LIB represents the library you are dumping in this run.
4.	This statement represents the sequential data set onto which the library is dumped. This data set must be large enough to contain the VISION:Transact library.
5.	The UC statement must be coded with the operation code DUMP to unload the library.

M5UTIL Restore

The operation code REST is used to restore a library from a backup copy. You have the ability to purge expired items during the restore process by coding a Y in position [50] of the UC statement. The library to be restored may be specified in the DD/DTF name field of the UC statement. If the ddname is not specified, M5LIB will be restored. [Figure 8-5](#) and [Figure 8-6](#) show the JCL necessary to restore BDAM and VSAM libraries.

For a VSAM library, you should dump the library ([Figure 8-4](#)), delete the cluster and define another cluster ([Defining a VSAM M5LIB on page 8-12](#)), and then restore the library ([Figure 8-6](#)).

```
//jobname JOB . . .
//*
//* THIS JCL IS USED TO RESTORE A BDAM M5LIB
//*
1> //rest EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel178.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.bdam.M5LIB,DISP=OLD
4> //M5WORK DD DSN=your.dump.M5WORK,DISP=SHR
//M5INPUT DD *
5> UCREST
//
```

Figure 8-5 Restore a BDAM Library

```
//jobname JOB . . .
//*
//* THIS JCL IS USED TO RESTORE A VSAM M5LIB
//*
1> //rest EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel178.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.vsam.M5LIB,DISP=OLD
4> //M5WORK DD DSN=your.dump.M5WORK,DISP=SHR
//M5INPUT DD *
5> UCREST
//
```

Figure 8-6 Restore a VSAM Library

Notes	Explanation
1.	This statement invokes M5UTIL.
2.	The M5LIST statement defines the location of the system output device for this job.
3.	M5LIB represents the library that will receive the backup sequential data set.

Notes	Explanation
4.	This statement represents the sequential backup data set you use to restore to M5LIB. It must have been created with a previous DUMP operation.
5.	The UC statement must be coded with the operation code REST to restore your library.

M5UTIL Condense

The operation code COND is used to condense a library. It removes the unused area in your library and concentrates free space that has been fragmented by library updating. COND dumps the usable items from your library to a sequential data set (M5WORK). It then restores the contents of the sequential data set to your library. The library to be condensed may be specified in the DD/DTF name field of the UC statement. If the ddname is not specified, M5LIB will be condensed. [Figure 8-7](#) and [Figure 8-8](#) show the JCL necessary to condense BDAM and VSAM libraries.

Note: For VSAM libraries, you should dump the library ([M5UTIL Dump on page 8-4](#)), delete its cluster, redefine the cluster ([Defining a VSAM M5LIB on page 8-12](#)), and then restore the library ([M5UTIL Restore on page 8-5](#)). This will ensure you receive meaningful statistics from VSAM (IDCAMS) as the high RBA is not reset when the library is condensed.

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO CONDENSE A BDAM M5LIB
/*
1> //cond EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.bdam.M5LIB,DISP=OLD
4> //M5WORK DD DSN=your.dump.M5WORK,DISP=(NEW,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(n,n),RLSE)
//M5INPUT DD *
5> UCCOND
//

```

Figure 8-7 Condense a BDAM Library

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO CONDENSE A VSAM M5LIB
/*
1> //cond EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.vsam.M5LIB,DISP=OLD
4> //M5WORK DD DSN=your.dump.M5WORK,DISP=(NEW,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(n,n))
//M5INPUT DD *
5> UCCOND
//

```

Figure 8-8 Condense a VSAM Library

Notes	Explanation
1.	This statement invokes M5UTIL.
2.	The M5LIST statement defines the location of the system output device for this job.
3.	M5LIB represents the library you are to condense in this run.
4.	This statement represents the sequential data set onto which the library is dumped. This data set must be large enough to contain your library.
5.	The UC statement must be coded with the operation code COND to condense your library.

M5UTIL Copy

The operation code COPY is used to copy cataloged items from one library to another. Cataloged items are copied from M5LIBn, where n=1-9, into M5LIB. On the UC statement, the items to be copied can be specified by full item name and item type, by generic item name, or just by item type (see the *Advantage VISION:Transact Specifications Guide*). If the ddname is not specified on the UC statement, items are copied from M5LIB1 into M5LIB.

Further qualification of items to be copied may be specified on the UC statement by using the Replace Option or Replace Rules fields. The Replace Option field specifies that items selected on the UC statement which already exist in M5LIB are to be replaced by like-named items copied from M5LIBn. If the replace option is not used, only those items not cataloged on M5LIB will be copied.

Replace Rules is qualified by coding either OLD or NEW. NEW specifies that the item on M5LIB will be replaced by a like-named item from M5LIBn if it has been updated more recently. OLD specifies that the item on M5LIB will be replaced by a like-named item from M5LIBn if it has been updated less recently.

Note: Copy operations with Replace Rules referencing cataloged definitions which do not have Item Tracking dates will fail and issue a message.

[Figure 8-9](#) and [Figure 8-10](#) show the JCL necessary to copy an item from a BDAM or VSAM library.

```
        //jobname  JOB . . .  
        //*  
        //*  THIS JCL IS USED TO COPY AN ITEM FROM A BDAM M5LIB  
        //*  
1>      //copy  EXEC PGM=M5UTIL,REGION=1M  
        //STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR  
2>      //M5LIST  DD SYSOUT=*  
3>      //M5LIB   DD DSN=your.bdam.M5LIB,DISP=OLD  
4>      //M5LIB1  DD DSN=your.other.bdam.M5LIB,DISP=SHR  
5>      //M5WORK  DD DUMMY  
        //M5INPUT DD *  
6>      UCCOPY   itemname  
        //
```

Figure 8-9 Copy a BDAM Library

```
        //jobname  JOB . . .  
        //*  
        //*  THIS JCL IS USED TO COPY AN ITEM FROM A VSAM M5LIB  
        //*  
1>      //copy  EXEC PGM=M5UTIL,REGION=1M  
        //STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR  
2>      //M5LIST  DD SYSOUT=*  
3>      //M5LIB   DD DSN=your.vsam.M5LIB,DISP=OLD  
4>      //M5LIB1  DD DSN=your.other.vsam.M5LIB,DISP=SHR  
5>      //M5WORK  DD DUMMY  
        //M5INPUT DD *  
6>      UCCOPY   itemname  
        //
```

Figure 8-10 Copy a VSAM Library

Notes	Explanation
1.	This statement invokes M5UTIL.
2.	The M5LIST statement defines the location of the system output device for this job.
3.	M5LIB represents the library that the cataloged item is copied into.
4.	M5LIB1 represents the library that the specified cataloged item is copied from.

Notes	Explanation
5.	An M5WORK file is required for all M5UTIL functions. Although required, it is not used by the COPY operation and can be dummied.
6.	The UC statement is coded with the operation code COPY to copy the cataloged item named "itemname" from M5LIB1 to M5LIB.

M5UTIL Merge

The operation code MERG is used to copy the entire contents of an M5LIBn into M5LIB, where n=1-9. The Library DD/DTF Name field of the UC statement is used to specify which M5LIBn will be merged into M5LIB; if it is not specified, M5LIB1 will be merged into M5LIB. The item name and type fields of the UC statement must be blank but the Replace Option and Replace Rules may be specified.

[Figure 8-11](#) and [Figure 8-12](#) show the JCL necessary to use the merge feature with BDAM and VSAM libraries.

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO MERGE TWO BDAM LIBRARIES
/*
1> //cond EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.bdam.M5LIB,DISP=OLD
4> //M5LIB1 DD DSN=your.other.bdam.M5LIB,DISP=SHR
5> //M5WORK DD DUMMY
//M5INPUT DD *
6> UC MERG
//

```

Figure 8-11 Merge a BDAM Library

```
        //jobname  JOB . . .
        //*
        //*  THIS JCL IS USED TO MERGE TWO VSAM LIBRARIES
        //*
1>      //cond    EXEC PGM=M5UTIL,REGION=1M
        //STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2>      //M5LIST  DD SYSOUT=*
3>      //M5LIB   DD DSN=your.vsam.M5LIB,DISP=OLD
4>      //M5LIB1  DD DSN=your.other.vsam.M5LIB,DISP=SHR
5>      //M5WORK  DD DUMMY
        //M5INPUT DD *
6>              UCMERG
        //
```

Figure 8-12 Merge a VSAM Library

Notes	Explanation
1.	This statement invokes M5UTIL.
2.	The M5LIST statement defines the location of the system output device for this job.
3.	M5LIB represents the library that cataloged items will be merged into.
4.	M5LIB1 represents the library that all cataloged items will be merged from.
5.	An M5WORK file is required for all M5UTIL functions. Although required, it is not used by the MERG operation and can be dummied.
6.	The UC statement must be coded with the operation code MERG to copy the entire contents of one library to M5LIB.

M5UTIL and Multiple UC Statements

[Figure 8-13](#) illustrates how multiple UC statements can be used with the M5UTIL program. For generality, each UC statement specifies a different operation code. Notice that this technique uses only one job step.

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO PERFORM MULTIPLE LIBRARY OPERATIONS
/*
1> //util EXEC PGM=M5UTIL,REGION=1M
//STEPLIB DD DSN=transact.rel178.GENLIB,DISP=SHR
//M5LIST DD SYSOUT=*
//M5LIB DD DSN=your.bdam.M5LIB,DISP=SHR
//M5LIB1 DD DSN=your.other.bdam.M5LIB,DISP=OLD
//M5LIB2 DD DSN=your.otherx.bdam.M5LIB,DISP=OLD
//M5LIB5 DD DSN=your.others.bdam.M5LIB,DISP=OLD
//M5LIB9 DD DSN=your.otherz.bdam.M5LIB,DISP=OLD
//M5WORK DD DSN=your.M5WORK,DISP=(NEW,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(1,1))
//M5INPUT DD *
2> UCINIT
3> UCCOND M5LIB2
4> UCMERG M5LIB9
5> UCCOPY FDfiledef M5LIB5
6> UCCOPY SDscreen
//

```

Figure 8-13 M5UTIL and Multiple UC Statements

Notes	Explanation
1.	M5UTIL processes the source statements from the M5INPUT data set.
2.	M5LIB is initialized. Since the ddname specification on the UC statement has no entry, the default library to be initialized is M5LIB.
3.	M5LIB2 is condensed.
4.	The entire contents of M5LIB9 is merged into M5LIB.
5.	The file definition “filedef” is copied from M5LIB5 to M5LIB.
6.	The screen definition named “screen” is copied from M5LIB1 to M5LIB. Since no sending library is specified on the UC statement, the default sending library is M5LIB1.

Note: For VSAM libraries, the JCL in [Figure 8-13](#) may need to be modified to include a JOBCAT DD statement.

Other Utility Programs

This section supplies JCL examples of utilities associated with the VISION:Transact library. The first example shows how you define a VSAM VISION:Transact library, using IDCAMS. The remaining examples show how you initialize, dump, restore, or condense a VISION:Transact library.

Defining a VSAM M5LIB

If your version of VISION:Transact supports VSAM M5LIBs, each M5LIB must be defined as a VSAM data set. [Figure 8-14](#) shows sample JCL statements for defining a VSAM M5LIB.

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO ALLOCATE A VSAM M5LIB CLUSTER
/*
//define EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER -
1>      (NAME(your.vsam.M5LIB) -
2>      VOLUME(xxxxxx) -
        CYL(1 1) -
        NUMBERED -
        SHAREOPTIONS(3 3) -
3>      RECORDSIZE(507 507) -
4>      CISZ(4096))
//
```

Figure 8-14 Sample JCL for Defining a VSAM M5LIB

Notes	Explanation
1.	"markv.m5lib" is the name of your M5LIB.
2.	"xxxxxx" is the volume serial number of the disk on which your M5LIB resides.
3.	The minimum RECORDSIZE allowable is 507.
4.	The 4096 CISZ gives you efficient processing in most cases.

The directory blocking factor (number of directory detail records in a DDR) is determined for VSAM M5LIBs by the record size specified in the cluster definition. A 507 byte record size means that all records on the M5LIB are 507 bytes. A 507 byte DDR allows 15 directory entries (each entry is 32 bytes) of which 12 point to data records or items. A larger record size increases the directory blocking factor; each additional 32 bytes will increase the directory entries by 1. The greater the blocking factor, the fewer accesses there will be to locate an item on M5LIB.

The “best” record size is the one that is closest to the average size library item and which fits without excessive waste in the chosen control interval size (CI size). You should also keep in mind the directory blocking factor the record size allows. It is advisable to choose a CI size that best balances a blocksize which is adequately large and data transfer time that does not stall the processing throughput. CI sizes of 2048 (2K) and 4096 (4K) are good CI sizes. VSAM is likely to compute a CI size of 2048 if not overridden in the cluster.

With a record size of 507 and CI size of 2048, four records will be stored in the CI. Each CI contains VSAM control information; a four-byte CIDE (one per CI) and a three byte RDF (one per record). Thus, four records in the CI is 2028 bytes (four times 507), plus 16 bytes of VSAM control fields (four for the CIDE and 12 for the RDFs). So with the 2048 CI size and 507 record size, 2044 bytes of the CI will be used and four bytes unused. The CI size and record size can best be decided after discovering the average size of your library items which will vary from one VISION:Transact installation to another. The average can be determined from a dump of an existing M5LIB. Directory records are easily spotted and bytes one through four of each directory detail entry contains the binary length of the data item. For a VSAM M5LIB, the length of a data item can exceed both record and CI length. Optimally, however, the record size and longest such data item will not exceed the CI size.

Use shareoptions of (3 3) unless you intend to access the library cluster with software other than VISION:Transact (such as, IDCAMS REPRO); if so, use (2 3).

Library Initialization Utility

Before file and screen processing definitions and requests can be cataloged, the VISION:Transact library must be initialized with a program called M5INIT. JCL statements to invoke M5INIT are shown in [Figure 8-15](#). Normally, this program is executed once. It allocates the required space and formats the area used for the library. Use this utility with caution because its inadvertent use on the same data set name will result in erasing previously cataloged items.

Note: For BDAM libraries the space must be contiguous.

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO ALLOCATE AND INITIALIZE AN M5LIB
/*
//init EXEC PGM=M5INIT,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
//M5LIST DD SYSOUT=*
//M5LIB DD DSN=your.M5LIB,
1> // DISP=(NEW,CATLG),
1> // UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
//

```

Figure 8-15 JCL Statements for M5INIT

Notes	Explanation
1.	If the data set already exists or if it is a VSAM library, replace these numbered statements with DISP=OLD. If you have VSAM M5LIBs, you may also need to include a JOBCAT DD statement to point to your VSAM catalog.

Library Condense Utility

The program M5CON is a VISION:Transact utility routine used to remove the empty space from the VISION:Transact library. It condenses the library by freeing areas from which items were previously deleted. M5CON requires a sequential work file with the ddname M5WORK. M5CON dumps the VISION:Transact library to a sequential data set specified by M5WORK, and then restores the condensed sequential data set to the new library. It automatically produces a listing which lists the name of each cataloged entry and its type.

Note: For VSAM libraries, you should dump the library ([Library Dump and Restore Utilities on page 8-16](#)), delete its cluster, redefine the cluster ([Defining a VSAM M5LIB on page 8-12](#)), and then restore the library ([Library Dump and Restore Utilities on page 8-16](#)). This will ensure you receive meaningful statistics from VSAM (IDCAMS) as the high RBA do not reset when the library is condensed.

The typical condense utility JCL statements are illustrated in [Figure 8-16](#).

```

//jobname JOB . . .
//*
/* THIS JCL IS USED TO CONDENSE AN M5LIB
/*
1> //init EXEC PGM=M5CON,REGION=1M
//STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2> //M5LIST DD SYSOUT=*
3> //M5LIB DD DSN=your.M5LIB,DISP=OLD
4> //M5WORK DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//

```

Figure 8-16 JCL Statements for M5CON

Notes	Explanation
1.	M5CON is the program required for condensing the VISION:Transact library.
2.	This M5LIST statement is used to specify the data set for outputting the output list.
3.	This M5LIB statement is used to define the VISION:Transact data set to be condensed. Note that the dictionary must be on a direct access device.
4.	This M5WORK statement is used to define a sequential work file. In this example, it is temporary.

Library Dump and Restore Utilities

Two additional VISION:Transact utility programs, M5DUMP and M5REST, together can simulate a M5CON run. They can be used to provide a backup for a VISION:Transact library or to copy a VISION:Transact library from one area to another.

The M5DUMP and M5REST JCL statements are illustrated in [Figure 8-17](#).

```
        //jobname  JOB . . .
        //*
        //*   THIS JCL IS USED TO DUMP & RESTORE AN M5LIB
        //*
1>      //init    EXEC PGM=M5DUMP,REGION=1M
        //STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
2>      //M5LIST  DD SYSOUT=*
3>      //M5LIB   DD DSN=your.M5LIB,DISP=OLD
4>      //M5WORK  DD DSN=&m5dump,
        //          DISP=(NEW,PASS),
        //          UNIT=SYSDA,SPACE=(CYL,(1,1))
        //*
5>      //init    EXEC PGM=M5REST,REGION=1M
        //STEPLIB DD DSN=transact.rel78.GENLIB,DISP=SHR
        //M5LIST  DD SYSOUT=*
6>      //M5LIB   DD DSN=your.new.M5LIB,
        //          DISP=(NEW,CATLG),
        //          UNIT=SYSDA,SPACE=(CYL,2,,CONTIG)
7>      //M5WORK  DD DSN=&m5dump,DISP=(OLD,DELETE)
        //
```

Figure 8-17 JCL Statements for M5DUMP and M5REST

Notes	Explanation
1.	M5DUMP is the program required to dump the VISION:Transact library.
2.	This M5LIST statement is used to specify the data set for outputting the output list.
3.	This M5LIB statement identifies the VISION:Transact library to be dumped.
4.	This M5WORK statement identifies the sequential data set onto which the library is to be dumped. This data set must be large enough to contain the VISION:Transact library.

Notes	Explanation
5.	M5REST is the program required to restore the VISION:Transact library. For a VSAM M5LIB, you might want to delete and redefine the cluster definition before the restore step. In that way the VSAM statistics from IDCAMS will be more meaningful.
6.	This M5LIB statement identifies the data set (on a direct access device) to which the library is to be restored. For a VSAM M5LIB, only the DSN and DISP=OLD should be specified.
7.	This M5WORK statement identifies the sequential data set that was created by an M5DUMP run.

Libraries On Shared DASD

There are different levels of protection for libraries when the libraries are being shared by multiple systems. The following describes the levels of protection for VSAM and BDAM libraries.

Your product issues a global ENQ macro instruction whether its RESERVE bit is on or off in the COMLIB installation parameters module MARKLIBP. This protects the library from concurrent updates from a single system (CPU).

If you have a COMLIB on shared DASD, the RESERVE option in MARKLIBP can be used to protect the COMLIB against concurrent updates from multiple systems (CPUs). When the MARKLIBP RESERVE bit is on, the system issues a hardware “lock” of the DASD where the COMLIB resides. This reserves a device for use by a particular system and protects against concurrent updates from multiple systems.

VISION:Transact uses a QNAME of M4LIB and an RNAME the same as the cluster or data set name when performing ENQs.

Global Resource Serialization (GRS), a component of MVS, enables your installation to share symbolically-named resources either between units of work in a single system or between multiple systems in a loosely coupled environment. When you have GRS, you do not need RESERVE because the local ENQ issued by the product provides sufficient protection. [Figure 8-18](#) shows the RESERVE and SHARE option protection for VSAM and BDAM COMLIBs.

		BDAM	VSAM
RESERVE	ON	GLOBAL ENQ & RESERVE	RESERVE & GLOBAL ENQ
	OFF	GLOBAL ENQ	GLOBAL ENQ

Figure 8-18 RESERVE and SHARE Option Protection for VSAM and BDAM COMLIB

Note: GLOBAL ENQ is only local in scope (one CPU) unless your installation has GRS.

The advantage of GRS over the RESERVE macro alone is that, instead of locking all the DASD, GRS issues a global ENQ macro instruction to serialize only the particular data set to be protected.

Note: In all cases, QNAME = M4LIB and RNAME = DSN

Reading VISION:Transact Application Dumps

Introduction

The purpose of this appendix is to help you locate application-related data areas (such as working storage, user flags, and data base segments) in a storage dump. The information presented here can also be used in conjunction with interactive debugging aids (such as CEDF under CICS TS or the TEST command under TSO) to locate significant data areas in an executing application.

In order to use the information presented here, you will need to have available an assembly listing of the VISION:Transact internal data structures. This assembly listing provides the layout of the individual data structures discussed in this appendix. The *Advantage VISION:Transact Installation Guide* contains instructions for producing this listing.

Overview of Data Areas

M5COM

This is the main internal communication area for a VISION:Transact generated application. The addresses of most of the other internal data structures are stored here. Register 11 should always point to M5COM.

GOODSTUF

This is another global work area. This area contains constants and work areas used by the data conversion routines. It also contains some screen related information. CVGOODST in M5COM contains the address of GOODSTUF.

DBDATBL

DBDATBL contains general information describing a particular data base or file, such as the file type, the address of the PCB, and the root PREFABL address. The DBDATBL address for a particular file is in M5COM at CVARUFBL plus the offset given in [Figure 0-1](#).

Since there is a separate SEGTABLE and DBDATBL for each file, they are addressed by tables of pointers in M5COM. The table of SEGTABLE addresses begins at CVPRSORG. DBDATBL addresses start at CVARUFBL. The same indexing scheme is used for both tables.

<u>File Type</u>	<u>Table Offset</u>	<u>File Type</u>	<u>Table Offset</u>
FLAGS	X'04'	FILE2	X'5C'
TEMPORARY FIELDS	X'08'	FILE3	X'60'
CONSTANTS	X'0C'	FILE4	X'64'
WORKING STORAGE	X'10'	FILE5	X'68'
PCBs (LINKAGE)	X'14'	FILE6	X'6C'
INPUT SCREEN	X'24'	FILE7	X'70'
FILE (PRIMARY)	X'50'	FILE8	X'74'
FILE1	X'58'	FILE9	X'78'

Figure 0-1 Table Offsets for DBDATBL and DBDATBLSEGTABLE Pointers

SEGTABLE

The SEGTABLE is used to supply general information about each segment type, such as length, level, and number of occurrences. It also contains information that controls automatic looping. There is a SEGTABLE for each data structure (such as, qualifier). There is a SEGTABLE for temporary fields, constants, working storage, linkage section (PCBs), flag fields, the current input screen, and for each data base or pseudo file.

The current accessibility of a segment type is indicated by SGFLAGS3. If either bit 1 (screen-matched) or bit 3 (loop in progress) is set, then SGPREF points to the PREFABL for the current occurrence, and SGSEGADR points to the segment itself.

A SEGTABLE is composed of one or more entries. There is one entry for each segment type in the structure to which it is related. For unstructured data, such as temporary fields and working storage, there is only one entry.

The address of the root (or only) SEGTABLE entry for a structure is in M5COM at CVPRSORG plus the offset given in [Figure 0-1](#). The subsequent entries for the same structure are in contiguous storage, arranged in top-to-bottom, left-to-right hierarchical sequence. Each SEGTABLE entry is 104 bytes long.

The relative position of a SEGTABLE entry is also known as the segment index. This is used within other internal data structures (such as, PREFABL, below) to identify segment type.

PREFABL

The PREFABL is used to describe the hierarchical relationships of each segment occurrence. There is one PREFABL for each memory-resident segment occurrence. There may also be “place holder” PREFABLs that represent missing occurrences (when no segments of a given type exist). Each PREFABL contains, among other things, four pointers:

1. PRPARENT points to the PREFABL of the parent segment.
2. PRNEXT points to the next twin or sibling PREFABL.
3. PRCHILD points to the first child PREFABL.
4. PRSEGADR points to the segment itself.

[Figure 0-2](#) and [Figure 0-3](#) illustrate how the PRPARENT, PRNEXT, and PRCHILD pointers are used to form the segment hierarchy. Since twin and sibling occurrences are intermixed on the PRNEXT chain, the field PRSEGNDX (segment index) is used to distinguish between them.

The PREFABL also contains a pointer to the SKEYHDR (described later) and the segment action code (PRACTION) associated with the segment occurrence.

The field DDROOT in a file's DBDATBL points to the first root PREFABL for that file. The current PREFABL is pointed to by SGPREF in the SEGTABLE.

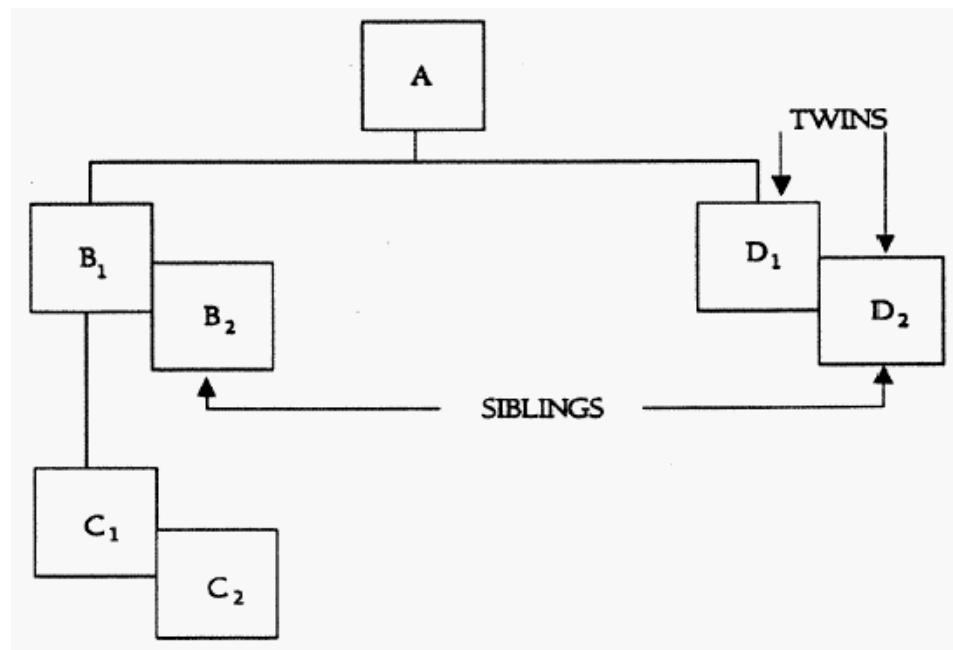


Figure 0-2 Logical Organization of a Data Base Record

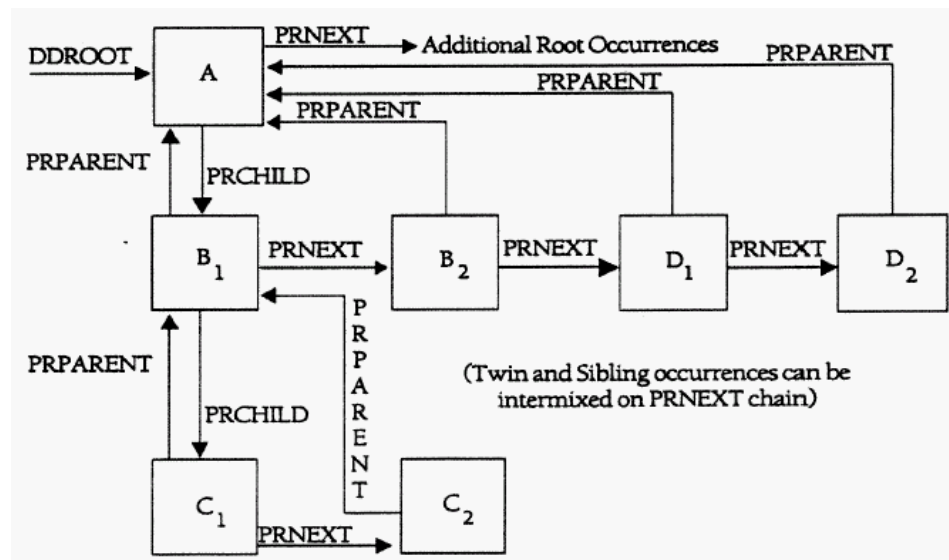


Figure 0-3 PREFABL Structure for the Same Data Base Record

SKEYHDR

This is the header for a list of keys and/or search fields for a segment. It exists for each qualified (such as, screen-matched or root DSDB) segment occurrence in core.

SKEYBL

The SKEYBL contains the actual key value used to retrieve or write the segment occurrence. A root segment in a browse file has two SKEYBL entries (to indicate the start and end of the browse range). Entries are chained via SKKNXT. SKHFST in the SKEYHDR points to the first SKEYBL entry for a segment.

DBWORKBL

The DBWORKBL is a common work/communication area for the VISION:Transact data base interface routines. It contains information related to the function in process, such as the file or data base name, file type, function to be performed, etc. The address of this area is in M5COM at CVDBWKBL.

TPDATBL

The TPDATBL contains information used in the interface with the teleprocessing monitor. There are three TPDATBL entries, corresponding to the primary, alternate, and error destination terminals, respectively.

The first entry contains the address of the IOPCB in IMS TM and is simply a place holder in CICS. The second TPDATBL entry contains the address of the modifiable TP PCB (IMS TM) or the name of the last-used alternate destination terminal (CICS). The third entry contains the address of the PCB for the error destination terminal (IMS TM) or the name of the error destination terminal (CICS).

These three TPDATBL entries are in contiguous storage, in the order listed above. Each entry is 8 bytes long. GOTPDATBL in GOODSTUF points to the first entry.

USERFLAG

This segment contains all the VISION:Transact user flag fields. The address of this area is in CVFLGBUF in M5COM.

Techniques

Locating M5COM

The first step in locating any application-related data is to locate M5COM, since it contains the pointers to all the other areas. Normally, Register 11 contains the address of M5COM. It can also be identified visually by means of the character constant "M5COM+16 *****" which is always found at offset 16 (decimal) from the start of M5COM.

Locating Working Storage

If the application uses working storage (i.e., contains RF statements with a file type of WORKnn), then CVWRKBUF in M5COM contains the address of the start of working storage. If multiple working storage records are defined, they are concatenated in numerical (WORKnn) sequence without any intervening slack bytes. See [Figure 0-4](#).

Locating Flag Data

CVFLGBUF in M5COM points to the area which contains all of the VISION:Transact user flags. See [Figure 0-4](#). This area is mapped by the USERFLAG DSECT.

Locating Temporary Fields

CVTMPBUF in M5COM points to an area containing the temporary fields and constants. If, on your APPGEN run control, you specify G for Source Statement Listing, a JD01 message is printed for each temporary field and constant giving its relative location within this area.

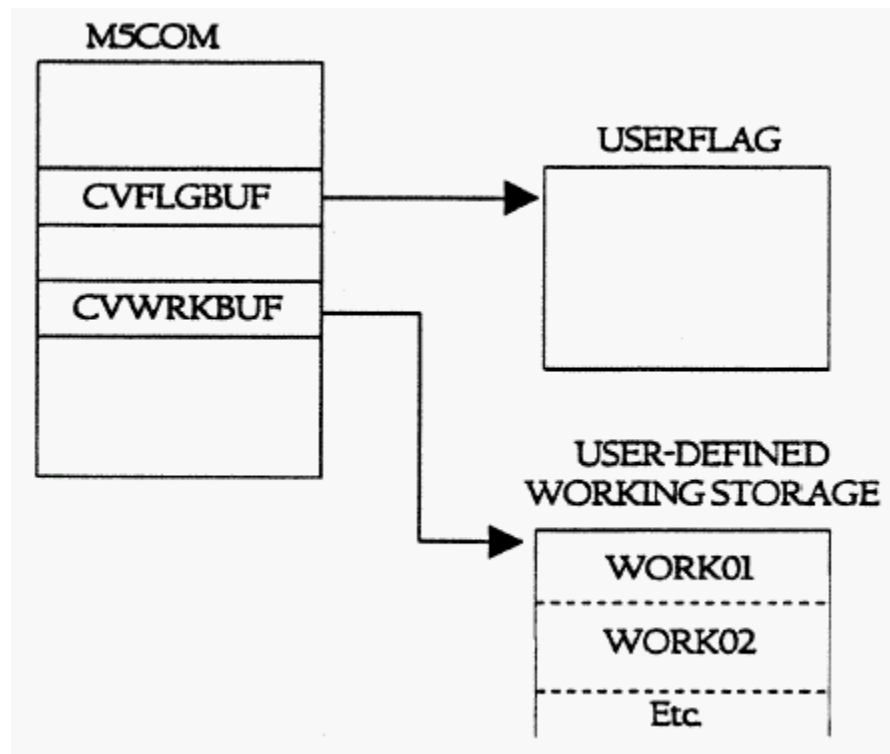


Figure 0-4 Location of Working Storage and User Flags

Locating Data Base Segments

VISION:Transact manages data base segments in main storage by means of the DBDATBL, SEGTABLE, and PREFABL structures described previously. The relationship between these structures is illustrated in [Figure 0-5](#).

To locate any or all in-memory segment occurrences for a given data base, first locate the DBDATBL as follows: Add the appropriate offset value from [Figure 0-1](#) to the address of CVARUFBL in M5COM. This gives you the address of a pointer to the DBDATBL.

Once you have located the DBDATBL, use the DDROOT pointer to locate the first root PREFABL. PRSEGADR points to the data for that segment occurrence. You can now follow the PRCHILD and PRNEXT chains to locate other segment occurrences (see [Figure 0-3](#)).

To locate the current occurrence of a particular segment, first locate the SEGTABLE by adding the appropriate offset value from [Figure 0-1](#) to the address of CVPRSORG in M5COM. This gives you the address of a pointer to the root SEGTABLE entry. Non-root entries follow the root in contiguous storage in top-down, left-to-right order. The field SGPREF points to the current PREFABL for each segment type.

The method for locating segment key information is illustrated in [Figure 0-6](#). PRKEYHDR in the PREFABL contains the address of an SKEYHDR, which in turn points to a chain of one or more SKEYBLs. The key value is a variable-length field in the SKEYBL.

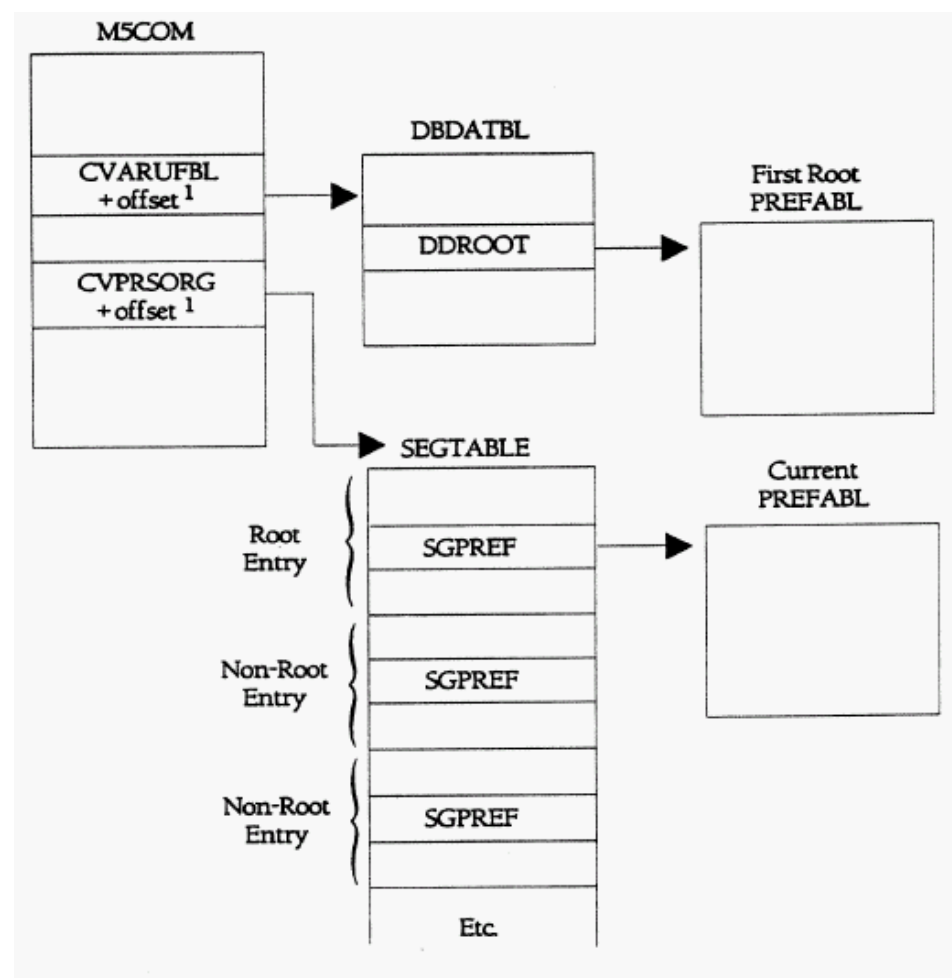


Figure 0-5 DBDATBL, SEGTABLE, and PREFABL Relationships

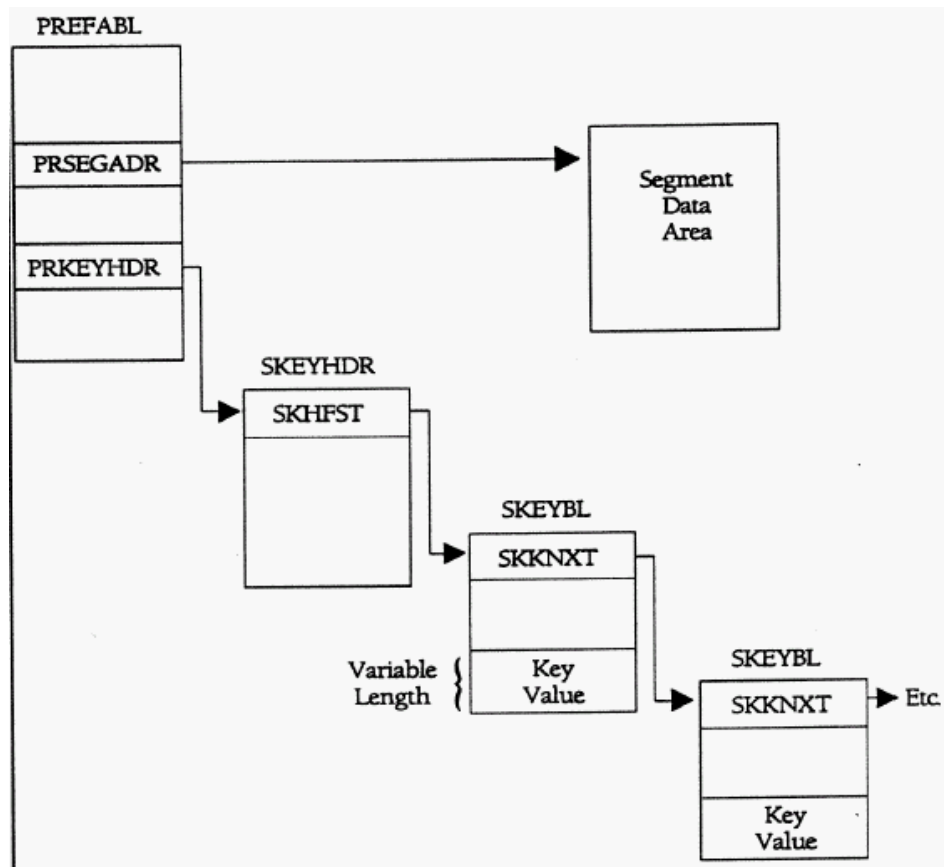


Figure 0-6 Relationship of PREFABL to Segment Data and Key Information

<u>Label</u>	<u>DSECT</u>	<u>Location (HEX)</u>
CVARUFBL	M5COM	628
CVDBWKBL	M5COM	CD8
CVFLGBUF	M5COM	55C
CVGOODST	M5COM	CD4
CVPRSORG	M5COM	7D8
CVTMPBUF	M5COM	560
CVWRKBUF	M5COM	568
DDROOT	DBDATBL	3C
GOTPD_TBL	GOODSTUF	984
PRCHILD	PREFABL	8
PRKEYHDR	PREFABL	14
PRNEXT	PREFABL	4
PRPARENT	PREFABL	0
PRSEGADR	PREFABL	C
SGPREF	SEGTABLE	3C
SKHFST	SKEYHDR	0
SKKNXT	SKEYBL	0

Figure 0-7 Summary of Pointer Location

Index

Symbols

?T statements, 4-4, 4-9, 6-4, 6-8

Numerics

3270 device, 7-24

3270 display devices and printers, 7-16

3270-type terminals, 6-11

A

AA statements, 4-4, 4-9, 8-2

ABEND, 6-10, 7-13

ABEND codes, 1-1

ACB, 5-1, 5-6

Alternate destination, 5-11

Alternate destinations, 7-24

Alternate library, 3-6, 4-12

Alternate terminal, 7-24

APPGEN, 2-2, 2-3, 4-1, 5-1, 5-4, 5-8, 6-11, 7-10, 7-24

APPLCTN macro, 4-5, 4-11, 5-5, 7-1

Application Control Block

ACB, 5-1

Application dumps, A-1

Application execution, 2-4

Application execution messages, 6-10

Application for multiple targets, 7-26

Application generation

APPGEN, 2-2

Application installation, 1-2, 2-3

Application link edit, 5-4

Application plan

Plan, 4-10

Application termination, 6-6, 7-25

ASMTDLI, 7-12

Assembler, 7-12, 7-21

ASSIGN command, 7-23

ATTNID flag, 7-17, 7-23, 7-25

Attributes, 7-17

AUTH parameter, 5-10

Authorization ID, 5-10

Auxiliary temporary storage, 7-19

B

Backout data base updates, 7-8, 7-11

Backup a library, 8-4, 8-16

Basic Mapping Support

BMS, 2-2

BIND, 5-10, 5-12

Binding an application plan, 5-1, 5-12

BMS, 2-2, 2-3, 3-1, 3-3, 3-4, 4-1, 4-11, 5-2, 5-3, 6-6

features supported, 7-24

- map data record, 7-23
- mapset, 2-3, 3-4, 5-3, 5-5, 5-8, 5-10, 6-6, 6-11, 7-23, 7-24, 7-26
- paging, 7-26
- paging commands, 7-24, 7-27
- routing, 7-24
- Browse, 7-9
- Browse DSDB, 7-5
- Browse synchronization, 7-9
- BTS, 7-26
- Buffer reshow, 5-10, 6-10, 7-11, 7-17, 7-25
- Buffers, 7-10, 7-18

C

- CALL ASMTDLI, 7-12
- CALL format, 7-27
- CALL statements, 6-4, 7-12, 7-13, 7-21
- CALL strategy, 7-8
- Called subroutines, 6-4, 7-10, 7-18, 7-27
- Calling Assembly language subroutines, 7-13
- Calling COBOL subroutines, 7-14
- Calling PL/I subroutines, 7-14
- Cataloged requests, 2-2, 3-1, 4-10
- CEDF, 6-7, A-1
- CEMT, 6-7
- CICS, 7-26
- CICS/VS, 1-1, 2-4, 3-1, 4-5, 4-11, 5-5, 6-5
 - commands, 6-7
 - execution, 5-5
 - executor, 7-18
 - table updates, 5-8
- CICSKEY, 7-25
- Clear key, 6-6, 7-25, 7-26
- COBOL, 7-12, 7-14, 7-21
- COMLIB, 1-1, 2-2, 3-1, 4-5, 4-10
- Command codes, 7-8

- COMMAREA, 7-21
- Comment statements, 4-9, 8-2
- Concepts and Facilities Manual, 2-1, 3-1
- COND operation, 8-6
- Condense a library, 8-6, 8-14
- Condition codes, 1-1
- CONV parameter, 7-16
- Conversational applications, 6-3
- Conversational program, 7-10, 7-15
- Conversational transaction, 7-15
- Copy items in a library, 8-7
- COPY operation, 8-7
- CR statements, 4-4, 4-9, 4-12
- CSECT, 7-1, 7-12, 7-13, 7-21
- CSMT, 6-7
- CT statements, 3-1, 3-6, 3-7
- CVARUFBL, A-8
- CVDBWKBL, A-5
- CVFLGBUF, A-5
- CVFLGBUF>, A-6
- CVPRSORG, A-3, A-8
- CVTMPBUF, A-6
- CVWRKBUF, A-6

D

- Data areas
 - DBDATBL, A-2
 - GOODSTUF, A-1
 - M5COM, A-1
- Data base, 3-1, 7-6
- Data base backout, 7-8, 7-11
- Data base definition
 - File definition, 2-2
- Data base PCB
 - PCB, 4-5

Data base segment occurrence, 7-6

Data request module

DBRM, 5-13

Date-compiled, 7-2

DB2, 1-1, 2-3, 4-10, 5-8, 5-12, 7-9

DB2 data bases, 4-2, 4-7, 5-1, 5-4, 5-5, 5-7, 5-8, 5-9, 5-12, 6-3, 6-6, 7-2, 7-9

DBD name, 7-6, 7-27

DBDATBL, A-2, A-7, A-8

DBDATBL address, A-2

DBRM, 5-13

DBRMLIB, 5-4, 5-5

DBWORKBL, A-5

DCT, 5-6, 5-8

DDROOT, A-3, A-7, A-8

Default input transaction code, 7-15

Deferred program message switch, 7-15

DEFINE CLUSTER, 8-3, 8-12

Defining a VSAM M5LIB, 8-12

Definition run, 3-1, 3-6, 5-2, 5-6, 5-8, 5-10, 7-24

Definitions

File or Screen definition, 2-2

Design manual, 4-5, 4-11

DEV statements, 7-17

Device field, 7-18

Device table, 6-11, 7-16, 7-24

Device type, 7-16

DFHMDI macro, 7-24

DFHMSD macro, 7-24

DFLD, 7-17

DIF/DOF, 6-2, 7-16

Directory block factor, 8-13

Display any modification, 7-25

DL/I, 6-3, 6-6, 7-5, 7-8, 7-12, 7-21, 7-27

DL/I path call usage, 7-8

DLI statements, 7-12

documentation

Advantage VISION

Transact Installation Guide, 1-1

Transact Message Guide, 1-1

Transact Reference Guide, 1-1

Transact Specifications Guide, 1-1

DSDB, 7-5, 7-7, 7-9, 7-10, 7-18, A-5

DSN command, 4-10

DSNHASM procedure, 5-4, 5-5, 5-13

Dump a library, 8-4, 8-16

DUMP operation, 8-4

Duplicate key processing, 7-6

DYNAM, 7-14

Dynamic linkage, 7-12, 7-21

Dynamic transaction code, 7-17

Dynamically synchronized data base, 7-5

E

EC operator, 6-10

EDF, 7-26

Empty fields, 7-26

ENDJOB, 7-14

Entry point, 7-13, 7-21

ER statements, 3-3, 4-4, 4-9

ERRDEST, 5-11, 6-10

Error cycle, 5-10

Error destination, 4-5, 4-11, 5-11, 6-1, 7-28

Error handling code, 7-3

Error line, 6-10

Error messages, 6-10

Error screen, 5-11, 6-6, 6-10

ET operator, 6-10

Execution, 5-4

Execution trace messages, 6-10

Execution trace symbols, 7-2

Execution tracing, 5-6, 5-8, 7-1, 7-6, 7-8, 7-9

IMS/DC applications, 6-4
Executor, 1-2, 5-5
executor, 7-9
EXIT, 6-3
Exit transaction, 7-25
Exit type screen, 7-25
EXPRESS parameter in PCB, 6-3

F

F command code, 7-9
Fast Path, 2-1
FCT, 5-10
FCT name, 7-27
FD statements, 3-3, 4-4, 4-9
Field separation, 7-27
File backout, 5-10
File buffers, 7-10, 7-18
File control, 6-6, 6-10
File definition, 2-2, 3-1, 4-1, 4-5, 4-10
File identification, 7-27
Files available, 7-28
Flag data, A-6
FORMAT command, 2-4, 6-2, 7-4
Format definitions, 7-16

G

Generated CSECTs, 7-1
Generator, 2-2, 3-1, 5-1, 6-11, 7-1
Generic item name, 8-7
Get Next call, 7-8
Get Unique call, 7-8
Glossary, 2-2, 3-1, 4-5, 4-11
GN call, 7-8, 7-9
GOODSTUF, A-1

GOTPD_TBL, A-5
GRANT, 5-13
GU call, 7-8, 7-9

H

HOLD, 6-3

I

I/O PCB, 5-11, 7-6, 7-11
I/O specification, 7-5, 7-17
IDCAMS, 8-12, 8-14, 8-17
IKJEFT01, 4-10
Immediate message switch, 7-15, 7-16
IMS data bases, 5-1, 5-6, 5-8, 6-3, 6-6, 7-5, 7-8
IMS operator, 7-8, 7-11
IMS security violation, 7-15
IMS system definition
 APPLCTN and TRANSACT macros, 5-5
IMS system generation, 5-5
IMS/DC, 1-1, 2-4, 3-1, 4-5, 4-11, 6-2, 6-3, 6-5, 6-10, 7-9, 7-16
 commands, 6-3
 execution, 5-4
IMSGEN, 5-7, 5-9, 5-12
 see IMS system generation, 5-7, 5-9, 5-12
IMSVS.FORMAT library, 5-2, 5-3
In-storage updates, 7-3
INCLUDE control statements, 7-12
INIT operation, 8-3
Initial entry, 2-4, 6-2, 6-6, 7-3
initialize a BDAM library, 8-3
Initialize a library, 8-3, 8-13
Input message, 7-17
Input screen, 6-10, 7-25
Input transaction code, 7-15

Input validation, 7-3
Input validation messages, 6-10
Insert calls, 7-8
Interaction, 7-11, 7-15, 7-25
Interactive debugging aids
 CEDF (CICS/VS), A-1
 TEST (TSO), A-1
Intermediate structure, 7-5
Internal data structures, A-1
Interrupts, 7-13
ISAM, 6-6
ISRT call, 7-15
IT statements, 3-3, 4-4, 4-9
Item name, 8-7, 8-9
Item tracking, 8-7
Item type, 8-7, 8-9

J

JCL, 8-3
 APPGEN with DB2, 4-7
 APPGEN without DB2, 4-2, 4-7, 5-7, 5-9, 5-12
 assembly and link edit of BMS maps, 5-3
 backup a BDAM library, 8-4
 bind application for DB2, 5-12
 define a VSAM M5LIB, 8-12
 definition run, 3-3
 link edit a CICS/VS application without DB2, 5-5
 link edit of IMS/DC application without DB2, 5-4
 precompile, assembly, & link edit of a CICS/VS application with DB2, 5-5
 precompile, assembly, & link edit of an IMS/DC application with DB2, 5-4
 source statement retrieval run, 3-5

K

Key field, 7-8

KEYLEN, 4-5, 4-11

L

L0 statements, 3-3, 4-4, 4-9
LCHILD, 4-5, 4-11, 7-8
LIBOPNA, 4-2, 4-4, 4-9, 4-13
Library
 Sterling MARK V library, 2-2
Library listing operations, 3-7
Library open, 4-2, 4-4, 4-9, 4-13
Library options, 6-1
Library routines, 6-5
Link edit, 2-2, 2-3, 4-1, 5-1, 5-4, 7-1, 7-12
Linkage editor, 7-12
LKEDCALL Programs, 7-12, 7-21
LKEDCALL RP statement option, 7-12, 7-14
LKEDCIS, 7-21
LKEDCIS>, 7-21
LKEDIMS, 7-12
Ln statements, 3-3, 4-4, 4-9
Load module, 7-9, 7-12, 7-18, 7-21
Load module name, 5-13, 7-26
locating, A-6
LOG parameter, 5-10
Logical data base, 5-11
LR statements, 3-3
LS statements, 3-3, 4-4, 4-9
LTERM parameter in PCB, 5-11

M

M5COM, A-1, A-5, A-8
 CVFLGBUF, A-5, A-6
 CVGOODST, A-1
 CVPRSORG, A-8
 CVTMPBUF, A-6

- CVWRKBUF, A-6
 - locating M5COM, A-6
- M5CON, 8-14, 8-16
- M5DUMP, 8-16
- M5INIT, 8-13
- M5INPUT, 2-2, 2-3, 3-3, 3-6, 4-4, 4-9
- M5LIB, 2-2, 3-6, 4-5, 4-10, 4-12
- M5LIBn, 3-6, 4-12
- M5LIST, 2-2, 2-3, 3-4, 3-6, 3-7, 4-5, 4-11, 5-1, 5-6, 8-3, 8-4, 8-5, 8-7, 8-8, 8-10, 8-15, 8-16
- M5MAP, 2-3, 4-6, 4-11, 5-1
- M5PARM, 4-2, 4-4, 4-9, 4-13, 5-11, 6-3, 6-6, 6-10, 7-9, 7-12, 7-17, 7-21, 7-24, 7-26, 7-28
 - CICSKEY, 7-25
 - device table, 6-11, 7-16, 7-24
 - ERRDEST, 5-11, 6-10
 - LIBOPNA, 4-2, 4-4, 4-9, 4-13
 - XLOAD, 6-3, 7-9
- M5PREP, 2-3, 4-1, 4-10, 4-12, 5-1, 7-9
- M5PUNCH, 2-2, 2-3, 3-4, 4-11, 5-1
- M5REST, 8-16
- M5SSOUT, 3-6
- M5TPTBL, 2-3, 4-6, 4-12, 5-1, 5-6, 5-7, 5-9, 5-10, 5-12
- M5UTIL, 8-1, 8-5
 - condense, 8-6
 - copy, 8-7
 - dump, 8-4
 - initialization, 8-3
 - merge, 8-9
 - multiple UC statements, 8-11
- M5WORK, 8-3, 8-9, 8-10, 8-15, 8-17
- Main temporary storage, 7-19
- MANUAL, 4-5, 4-11
- MDT, 7-5, 7-17
- Merge a library, 8-9
- MERGE operation, 8-9
- Message format for IMS, 7-17
- Message line, 6-10

- Message processing, 7-26
- Message processing region
 - MP region, 6-2
- Message queue, 6-2
- Message switch
 - Program-to-program message switch, 6-3
- Messages, 1-1, 4-11, 6-10
- MFLD, 7-17
- MFS, 2-2, 2-3, 3-1, 3-3, 4-1, 4-6, 4-11, 5-2, 5-3, 6-2
 - features supported, 7-16
 - paging commands, 7-17, 7-27
- MID, 6-2
- Missing indicators, 7-15, 7-16
- MOD, 6-2, 7-4
- MODIFY parameter in PCB, 5-11
- MP region, 6-2
- MS operator, 6-10, 7-5, 7-25
- MSG command, 7-5, 7-25
- MULTILIB, 3-7, 4-12
- Multiple targets for an application, 7-26

N

- NAME link edit, 7-1
- Names of cataloged items, 3-1
- NCAL parameter, 7-12, 7-21
- New copy, 6-7
- NEXT REQUEST branch, 7-5
- Non-conversational program, 7-15, 7-16
- Null screen segment processing, 7-25

O

- Object code, 4-1, 4-5, 4-10, 7-1
- Object module, 2-3, 4-6, 4-11, 5-1, 5-4
- OC operator, 6-7, 7-15, 7-16, 7-18, 7-22, 7-24
- Online applications, 1-2, 4-1

Open files, 6-7
Operator Logical Paging, 7-17
Originating terminal, 6-2, 6-5, 6-6, 6-7, 6-10, 7-23
OS operator, 7-23
OUTLIM parameter, 7-18, 7-24, 7-27
Output control, 7-18, 7-24
Output message, 7-18
Output screen, 7-3, 7-16, 7-18, 7-22, 7-23
Overflow indicators, 7-15, 7-16

P

PA keys, 6-6, 7-25, 7-26
Paging commands, 7-17, 7-24, 7-27
Passing control to another CICS/VS transaction, 7-22
Passing control to another IMS/DC transaction, 7-15
Passing data between interactions, 7-25
Path calls, 7-9
Path insert calls, 7-8
PCB, 4-5, 4-11, 5-11, 6-3, 6-10, 7-1, 7-6, 7-27
PCB address, A-2
PCB masks, 3-1
PCB selection, 7-7, 7-27
PCT, 4-12, 5-6, 5-8, 5-10, 7-18
PF keys, 7-17, 7-26
PICA, 7-13
PL/I, 7-12, 7-14
PLAN name, 4-10, 5-10, 5-13
PLI statements, 7-12, 7-14
PLICALLA, 7-14
POS parameter in PCB, 5-11
Positional PCB, 7-7, 7-27
PPT, 3-4, 4-6, 4-12, 5-6, 5-8, 5-10, 7-21
PR statements, 3-3, 4-4, 4-9
PRACTION, A-3
PRCHILD, A-7
PRCHILD pointer, A-3
Precompiling and assembling SQL statements, 5-1
PREFABL, A-2, A-3, A-7, A-8
 PRCHILD pointer, A-3
 PRNEXT pointer, A-3
 PRPARENT pointer, A-3
 PRSEGADR pointer, A-3
PREFABL address, A-2
Primary library, 3-3, 3-6, 4-5, 4-10, 4-12
PRKEYHDR, A-8
PRNEXT chain, A-3
PRNEXT pointer, A-3
PRNEXT>, A-7
Processing options
 PROCOPT, 4-5
Processing Program Table
 PPT, 3-4
PROCOPT, 4-5, 4-11, 5-11
Program Control Table
 PCT, 4-12
Program name, 7-1, 7-26
Program Processing Table
 PPT, 4-12
Program Specification Block
 PSB, 2-3
Program-to-program message switch, 6-3, 7-15
PRPARENT pointer, A-3
PRSEGADR, A-7
PRSEGADR pointer, A-3
PRSEGNDX segment index, A-3
PSB, 2-3, 4-5, 4-6, 4-11, 4-12, 5-1, 5-6, 5-12, 7-8, 7-27
PSB name, 7-26
PSBGEN, 5-8, 7-6
Pseudo conversational processing, 7-19

Pseudo files, 3-1, 6-7, 7-10, 7-15, 7-18, 7-22, 7-23

Q

Q command code, 7-8, 7-9

R

RC statements, 3-3, 4-2, 4-4, 4-9, 4-13, 6-6, 7-1, 7-12, 7-21, 7-25, 7-26

RCT, 4-12, 5-6, 5-8, 5-10

Re-read record or segment, 7-5

RECYCLE option, 7-11

Redisplay input screen, 6-10, 7-5, 7-8, 7-11, 7-17, 7-25

Relational data bases

DB2 data bases, 4-2, 4-7, 5-7, 5-9, 5-12

RELEASE, 6-3

REPLACE option, 8-7, 8-9

Request name, 4-4, 4-9

Requests, 2-2, 2-3, 3-1, 3-6, 4-5, 4-10, 4-13, 6-4

RESIDENT, 7-14

Resource Control Table

RCT, 4-12

REST operation, 8-5

Restore a library, 8-5, 8-16

restore>, 8-5

RETRIEVE command, 7-23

Return code flag, 7-13

RETURN command, 7-21, 7-22

Return point, 7-13, 7-21

RETURNCD flag, 7-13

Reusable user routines, 7-13

RF statements, 4-1, 4-4, 4-9, 7-6, 7-8, 7-10, 7-15, 7-19, 7-22

ROLB call, 7-8, 7-11

Rollback, 7-8, 7-11

ROLLBACK sync point, 5-10

RP statements, 3-3, 3-7, 4-4, 4-9, 4-12, 7-12

RP statements <\$IINCLUDE control statements<\$ILKEDCALL RP statement option, 7-21

RTT, 5-13

Run control group, 4-4, 4-9

Run control statement

RC statements, 3-3

Run files

RF statements, 4-1

Run name, 3-3, 3-6, 4-9

Run parameter statement

RP statements, 3-3

Run type, 3-3, 3-6, 4-4, 4-9

S

S0 statements, 3-3, 4-4, 4-9, 7-5, 7-25

Save area, 7-13, 7-21

SC statements, 4-1, 4-4, 6-3, 6-6, 6-7, 7-4, 7-15, 7-18, 7-24, 7-27

Scratch Pad Area

SPA, 7-10

Screen buffers, 7-9, 7-18

Screen control, 4-1, 6-6, 7-2, 7-4, 7-15

Screen control branch, 7-25

Screen control statement

SC statements, 4-1

Screen definition, 2-2, 3-1, 4-5, 4-10, 5-2, 5-6, 5-10, 6-6, 6-10

Screen field location and attribute table, 7-2

Screen fields, 7-25

Screen format, 7-2

Screen formatting source statements

MFS for IMS/DC or BMS for CICS, 2-2

Screen height, 7-24

Screen identifier, 7-24, 7-27

Screen name, 6-2, 6-6, 7-17, 7-23
Screen processing definition
 Screen definition, 2-2
Screen record, 7-19
Screen synchronization data base
 SSDB, 7-5
Screen table, 7-2
Screen width, 7-24
SCS printer devices, 7-16
SD statements, 3-3, 4-4, 4-9, 7-15, 7-16, 7-18, 7-24, 7-27
Search field, 7-8
Secondary indexes, 5-11
Secondary library, 3-6, 4-12
Security violation, 7-15
Segment action, 7-8
Segment counters, 7-27
Segment index, A-3
Segment search argument, 7-8
Segment table, 7-2
SEGTABLE, A-2, A-7, A-8
SG SELECT ALL, 7-5
SGFLAGS3, A-2
SGPREF, A-2, A-3, A-8
SGSEGADR, A-2
Shareoptions, 8-13
SKEYBL, A-5, A-8
SKEYHDR, A-3, A-5, A-8
SKHFST, A-5
SKKNXT, A-5
SL statements, 3-3, 4-4, 4-9
Sn statements, 3-3, 4-4, 4-9, 7-5
Source statement listing, 2-2
Source statement retrieval run, 3-4
Source statements, 3-4, 4-5, 4-11
SPA, 7-10, 7-15, 7-26

SPIE macro, 7-13
SQL, 1-1, 2-3, 4-1, 4-10, 4-12, 5-1, 6-3, 6-6, 7-9
SS statements, 3-3, 4-4, 4-9
SSA, 7-9
 Segment search argument, 7-8
SSDB, 7-7, 7-8, 7-10, 7-18
 Screen synchronization data base, 7-5
Standard PCB selection, 7-6
Start a new program, 7-28
Start a transaction, 6-7
Start an application, 6-6
START command, 7-23
START operator, 7-28
START parameter, 6-7, 7-22
Static linkage, 7-12, 7-21
Sterling MARK V library, 4-5
Sterling MARK V library, 2-2, 3-1, 3-3, 3-6, 4-10, 4-12
Storage requirement for IMS applications, 7-9
Storage requirements for CICS applications, 7-18
Subroutine
 Called subroutines, 6-4
Synchronizing field, 7-9
Syncpoint processing, 7-9
Syncpoint rollback, 5-10
SYSLIB, 5-4, 5-5, 7-12, 7-21
System modification, 7-2
System rollback control, 7-8, 7-11

T

TAPPFLOW, 6-9
Target monitor, 7-26
Task, 6-6, 7-23
TCONTROL, 6-9
TDLOOP, 6-9
TDSDBIO, 6-9

Temporary fields, 7-2, 7-18, A-6
Temporary storage, 6-6, 6-8, 7-18, 7-24, 7-26
Terminal ID, 7-23
Terminal support, 6-11
Terminal table, 6-11
TEST (TSO), A-1
TF statements, 3-3, 4-4, 4-9
TILOOP, 6-9
Time-compiled, 7-2
TP monitor, 3-3
TP monitor definitions, 2-3, 4-6, 4-12
TP PCB
 PCB, 4-5
TPDATBL, A-5
 alternate destination terminal address, A-5
 error destination address, A-5
 IOPCB address, A-5
 TP PCB address, A-5
TPROC, 6-9
Trace flags, 6-4, 6-8
Trace messages, 6-10
Trace options, 6-4, 6-8
Trace output
 CICS/VS applications, 6-8
 IMS/DC applications, 6-4
TRANCODE flag, 6-3, 6-6, 6-7, 7-15, 7-17, 7-22, 7-25, 7-28
TRANSACT macro, 4-5, 4-11, 5-5, 5-7, 5-9, 5-12, 7-1
Transaction, 7-15
transaction, 7-16
Transaction code, 2-4, 6-2, 6-3, 6-6, 7-17, 7-27
Transaction identification, 6-6
Transaction initiation
 Initial entry, 2-4
Transaction name
 Transaction code, 2-4
TRANSID parameter, 7-22

Transient data, 5-6, 5-8, 6-8, 6-9
TSEGACTION, 6-9
TSO Terminal monitor program, 4-10
TSSDBIO, 6-9
TWA, 7-18, 7-21
Type 0 message, 3-7, 4-13
Type 1 request, 7-3
Type 2 request, 7-3
Type 3 request, 7-3
Type M request, 7-3
Type O request, 7-3, 7-15, 7-22, 7-25
Type S request, 7-3

U

Unexpected DL/I status code, 6-10
Unexpected return code, 6-10
USAGE=MAP, 5-10
User written subroutine
 Called subroutines, 6-4
USERFLAG, A-5
USERFLAG DSECT, A-6

V

VISION
 ransact library, 8-1
VSAM, 7-5
VSAM cluster, 8-3, 8-6, 8-12
VSAM control interval, 7-18
VSAM files, 6-6, 7-18

W

Working storage, 3-1, 6-7, 7-10, 7-15, 7-19, 7-22, 7-23
 locating, A-6

X

XIND parameter, 7-15, 7-16, 7-18

XLOAD, 6-3

