

CA-IDMS[®]

VSAM Transparency
15.0



Computer Associates™

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.

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.

First Edition, December 2000

© 2000 Computer Associates International, Inc.
One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

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

Contents

How to use this manual	vii
Chapter 1. Introduction to CA-IDMS/VSAM Transparency	1-1
1.1 Overview	1-3
1.1.1 Data access method conversion process	1-3
1.1.2 Integration with CA-IDMS/DB	1-5
1.1.3 Implementation	1-6
1.2 General architecture	1-7
1.3 Operating requirements	1-8
Chapter 2. CA-IDMS/DB Schema and VSAM File Correspondences	2-1
2.1 Overview	2-3
2.2 Key-sequenced data set	2-4
2.2.1 Record and set representations	2-4
2.2.2 Location mode options	2-4
2.2.3 Example of schema definition	2-5
2.3 Entry-sequenced data set	2-6
2.3.1 Record and set representations	2-6
2.3.2 Location mode options	2-6
2.3.3 Examples of schema definition	2-7
2.4 Relative-record data set	2-8
2.4.1 Record and set representations	2-8
2.4.2 Location mode options	2-8
2.4.3 Example of schema definition	2-9
2.5 Alternate index	2-10
2.5.1 Record and set representations	2-10
2.5.2 Location mode options	2-10
2.5.3 Example of schema definition	2-11
2.6 Schema definition summary	2-12
2.6.1 Schema definitions for VSAM data structures	2-13
2.7 Area and file considerations	2-14
Chapter 3. Record Access and Processing	3-1
3.1 Overview	3-3
3.2 Keyed access	3-4
3.2.1 Keyed direct processing	3-4
3.2.2 Keyed sequential processing	3-4
3.2.3 Keyed skip-sequential processing	3-5
3.2.4 Keyed backward processing	3-5
3.3 Addressed access	3-6
3.3.1 Addressed direct processing	3-6
3.3.2 Addressed sequential processing	3-6
3.3.3 Addressed backward processing	3-6
3.4 DML correspondences to VSAM commands	3-7
3.5 Additional VSAM Options	3-8
Chapter 4. Preparing Control Information	4-1

4.1	Overview	4-3
4.2	Compiler-directive statements	4-5
4.2.1	Example	4-6
4.3	File management table	4-7
4.3.1	Usage	4-10
4.3.2	Examples	4-12
4.3.3	FMT JCL	4-13
4.4	Transaction name table	4-17
4.4.1	Examples	4-18
4.4.2	TNT JCL	4-19
Chapter 5.	Runtime Operations	5-1
5.1	Overview	5-3
5.2	Database preparation	5-4
5.2.1	Step 1: Install and prepare the DC/UCF system	5-4
5.2.2	Step 2: Prepare and install the CA-IDMS/DB database	5-4
5.2.3	Step 3: Create and compile the control tables	5-5
5.2.4	Step 4: Migrate data to the CA-IDMS/DB database	5-5
5.2.4.1	Sample migration utility JCL — OS/390	5-5
5.2.4.2	Sample migration utility JCL — VSE/ESA	5-8
5.3	Application preparation	5-11
5.3.1	Step 1: Modify the application JCL	5-11
5.3.1.1	Modify the application JCL — OS/390	5-11
5.3.1.2	Modify the application JCL — VSE/ESA	5-18
5.3.1.3	CA-IDMS/VSAM Transparency parameters	5-23
5.3.2	Step 2: Modify and recompile the application program	5-26
5.4	System execution	5-27
5.4.1	Step 1: Run the CA-IDMS/VSAM Transparency command interface	5-27
5.4.1.1	CA-IDMS/VSAM Transparency system commands	5-27
5.4.1.2	CA-IDMS/VSAM Transparency initialization — OS/390	5-28
5.4.1.3	CA-IDMS/VSAM Transparency initialization — VSE/ESA	5-28
5.4.2	Step 2: Bring up the DC/UCF system	5-29
5.5	Application execution	5-30
5.5.1	Normal termination	5-30
5.5.2	Abnormal termination	5-30
5.5.2.1	CA-IDMS/VSAM Transparency shutdown procedures — OS/390	5-31
Appendix A.	CA-IDMS/VSAM Transparency Architecture	A-1
A.1	About this appendix	A-3
A.2	CA-IDMS/VSAM Transparency architecture — batch processing	A-4
A.3	System services manager	A-5
A.4	Request processing modules	A-6
A.4.1	The CA-IDMS/VSAM Transparency front end	A-6
A.4.2	The CA-IDMS/VSAM Transparency back end	A-7
A.5	CA-IDMS/VSAM Transparency architecture — CICS processing	A-8
A.6	Control tables	A-10
A.6.1	File management table	A-10
A.6.2	Transaction name table	A-10
Appendix B.	CA-IDMS/VSAM Transparency Installation	B-1
B.1	About this appendix	B-3

B.2 OS/390 Installation	B-4
B.2.1 Installing CA-IDMS/VSAM Transparency in the CICS environment	B-5
B.3 VSE/ESA Installation	B-7
B.3.1 Installing CA-IDMS/VSAM Transparency in the CICS environment	B-8
Appendix C. Variable-Length Record Considerations	C-1
C.1 About this appendix	C-3
C.2 Using variable-length CA-IDMS/VSAM Transparency records	C-4
Appendix D. CA-IDMS/VSAM Transparency Return Codes and Messages	D-1
D.1 About this appendix	D-3
D.2 Compiler messages	D-4
D.2.1 FMT compiler messages	D-4
D.2.2 TNT Compiler messages	D-7
D.3 Command interface messages	D-11
D.3.1 Command interface messages — OS/390	D-11
D.3.2 Command interface messages — VSE/ESA	D-13
D.3.3 Front-end messages — OS/390 and VSE/ESA	D-16
D.4 Run-time messages — OS/390	D-17
D.5 Run-time messages — VSE/ESA	D-18
D.6 Migration messages	D-19
D.6.1 Migration messages — OS/390	D-19
D.6.2 Migration messages — VSE/ESA	D-19
D.7 Run-time feedback codes	D-20
D.7.1 OPEN codes	D-20
D.7.2 CLOSE/TCLOSE codes	D-22
D.7.3 REQUEST macro codes	D-23
Appendix E. CA-IDMS/VSAM Transparency User Exits	E-1
E.1 About this appendix	E-3
E.2 COBOL user exit program	E-4
E.2.1 Program requirements	E-4
E.2.2 COBOL user exit control block	E-5
E.2.3 Sample COBOL user exit	E-11
E.3 Assembler user exit program	E-17
E.3.1 Program requirements	E-17
E.3.2 Assembler user exit control block	E-18
E.3.3 Assembler user exit template	E-21
E.3.4 Sample assembler user exit	E-22
E.3.5 Macros	E-29
Appendix F. TSO File Allocation with CA-IDMS/VSAM Transparency	F-1
F.1 About this appendix	F-3
F.2 TSO file allocation syntax	F-4
Appendix G. VSE/ESA CICS SYSESVS Parameters	G-1
G.1 About this appendix	G-3
G.2 SYSESVS parameter file	G-4
Appendix H. IDMSLBS Procedure	H-1

H.1 About this appendix	H-3
H.1.1 IDMSLBLS Procedure	H-3
Index	X-1

How to use this manual

What this manual contains

This document introduces the user to CA-IDMS/VSAM Transparency 15.0. It describes:

- Capabilities of CA-IDMS/VSAM Transparency
- Procedures necessary to use CA-IDMS/VSAM Transparency
- Instructions for installing CA-IDMS/VSAM Transparency

How this manual is organized

Chapter	Description
1	Describes the general purpose, design, and operating requirements of CA-IDMS/VSAM Transparency
2	Describes how to define VSAM data structures to the CA-IDMS/DB schema
3	Describes the access and processing options available to the CA-IDMS/VSAM Transparency user
4	Presents instructions and syntax for preparing and compiling the CA-IDMS/VSAM Transparency control tables
5	Describes the steps you follow to execute CA-IDMS/VSAM Transparency applications
A	Presents a detailed description of CA-IDMS/VSAM Transparency components
B	Describes the steps you follow to install CA-IDMS/VSAM Transparency in a CA-IDMS/DB environment
C	Describes the steps you follow to use variable-length records in CA-IDMS/VSAM Transparency
D	Describes the return codes and messages issued by the control table compilers and by the CA-IDMS/VSAM Transparency command interface
E	Describes the use of CA-IDMS/VSAM Transparency user exits in either COBOL or Assembler.
F	Describes the allocation of CA-IDMS/VSAM Transparency files under TSO
G	Describes the VSE/ESA CICS SYSESVS parameters which are read and processed when a VSE/ESA VSAM/T CICS interface is started.
H	Describes the IDMSLBLE procedure used in VSE/ESA environments to simplify the creation of VSE/ESA JCL job streams.

Related documentation

For further information related to this document see the following Computer Associates documents:

- CA-IDMS installation manual for your operating system
- *CA-IDMS System Generation*
- *CA-IDMS Database Design Guide*
- *CA-IDMS System Operations*
- *CA-IDMS Database Administration*

Chapter 1. Introduction to CA-IDMS/VSAM Transparency

- 1.1 Overview 1-3
 - 1.1.1 Data access method conversion process 1-3
 - 1.1.2 Integration with CA-IDMS/DB 1-5
 - 1.1.3 Implementation 1-6
- 1.2 General architecture 1-7
- 1.3 Operating requirements 1-8

1.1 Overview

What is CA-IDMS/VSAM Transparency?: CA-IDMS/VSAM Transparency is a program interface that allows VSAM-based applications to access and update a CA-IDMS/DB database. Through CA-IDMS/VSAM Transparency, users can execute the following against CA-IDMS/DB data:

- Batch VSAM programs
- CICS VSAM programs
- VSAM-based packaged software programs

Because CA-IDMS/VSAM Transparency supports all commonly used VSAM features, modification or recompilation of the VSAM application is generally not required.

CA-IDMS/VSAM Transparency provides an easy and efficient way to migrate from a non-CA-IDMS/DB file structure to CA-IDMS/DB. CA-IDMS/VSAM Transparency allows you to:

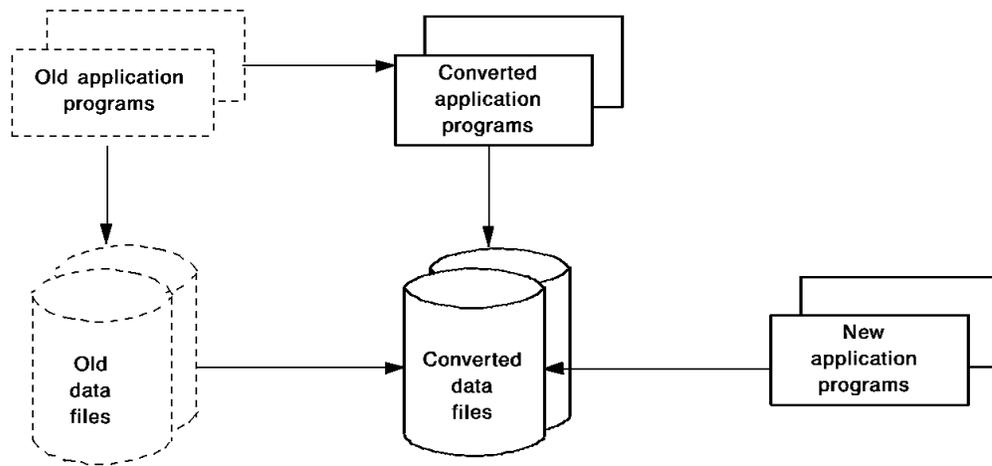
- **Run old VSAM applications and new CA-IDMS/DB programs against the same database.** This allows the conversion to CA-IDMS/DB technology to occur gradually and without loss of investment in existing application systems.
- **Run the same VSAM application against both the CA-IDMS/DB database and VSAM files.** This prevents the disruption of normal data processing procedures during the migration process.

About this chapter: This chapter provides introductory information on the following:

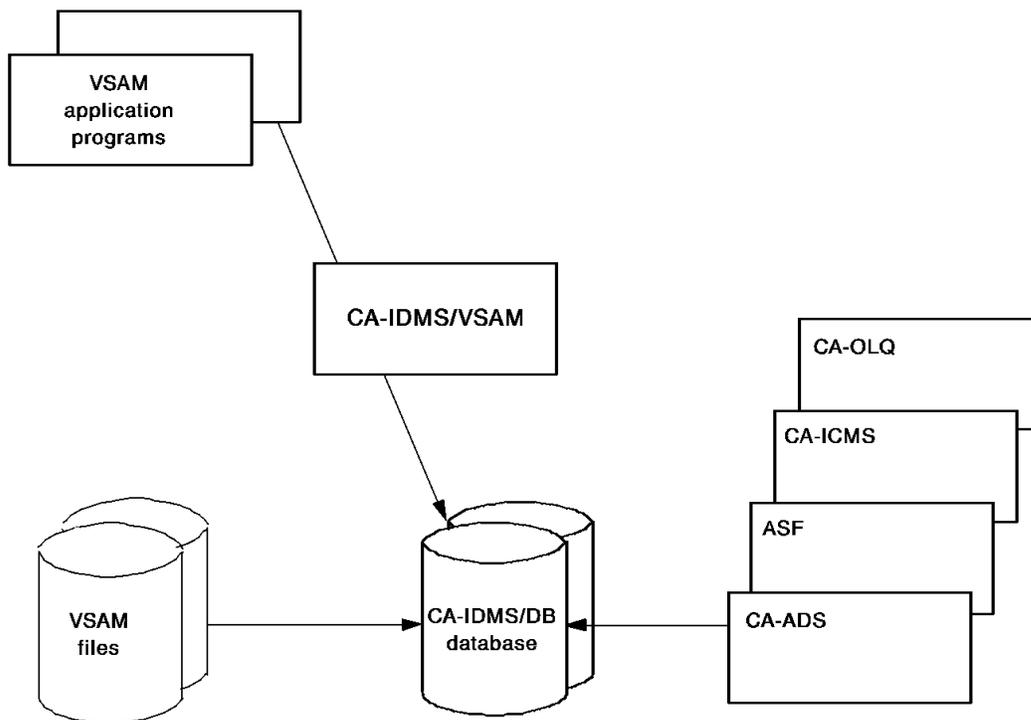
- How CA-IDMS/VSAM Transparency converts and integrates data to a CA-IDMS/DB format
- How CA-IDMS/VSAM Transparency processes requests using control information
- The general architecture of CA-IDMS/VSAM Transparency
- CA-IDMS/VSAM Transparency operating requirements

1.1.1 Data access method conversion process

Conventional access method: The following diagram illustrates the conventional access method conversion process. Typically, conversion from one data access method to another requires simultaneous conversion of application programs and data files.



CA-IDMS/VSAM Transparency access method: The following diagram illustrates the access method conversion process with CA-IDMS/VSAM Transparency. Existing VSAM applications can run as they are against CA-IDMS/DB files. Only the data must be converted to CA-IDMS/DB format, not the programs.



1.1.2 Integration with CA-IDMS/DB

CA-IDMS/VSAM Transparency is totally integrated into the regular CA-IDMS/DB environment. This integration allows access to the following database services and facilities:

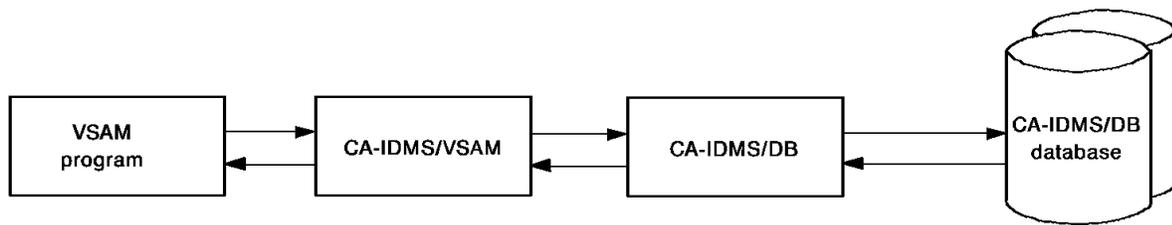
- CA-IDMS/DB central version (CV) and local mode operations
- Integrated data dictionary control
- Record locking
- Automatic recovery
- Journaling
- Batch application execution
- CICS online application execution
- Concurrent access to data by multiple programs that run simultaneously
- CA application development tools, such as CA-ADS, CA-OLQ, CA-ICMS, and the Automatic System Facility (ASF)

Translating requests: The interface between VSAM application programs and the CA-IDMS/DB database depends on user-defined control information. This information establishes a correspondence between the data viewed by the VSAM program and the database structures defined in the CA-IDMS/DB schema. At run time, CA-IDMS/VSAM Transparency uses the control information to translate VSAM processing requests into CA-IDMS/DB database calls.

Processing requests: Requests pass between the VSAM application and the database as follows:

1. CA-IDMS/VSAM Transparency accepts processing requests from the calling program.
2. CA-IDMS/VSAM Transparency converts processing requests to CA-IDMS/DB calls.
3. CA-IDMS/DB accesses the database and performs requested updates and retrievals.
4. CA-IDMS/VSAM Transparency returns status information and data (as appropriate) in VSAM format to the application program.

Flow diagram: The following diagram illustrates how CA-IDMS/VSAM Transparency processes VSAM requests against a CA-IDMS/DB database. CA-IDMS/VSAM Transparency acts as a translator between VSAM and CA-IDMS/DB, allowing VSAM application programs to access a CA-IDMS/DB database.



Database environment: The CA-IDMS/DB database used in the CA-IDMS/VSAM Transparency environment is a standard CA-IDMS/DB database that can be used by other CA-IDMS/VSAM Transparency and CA-IDMS/DB applications. Physical database details such as areas, record location modes, set orders, and indexing are transparent to the application program. You can change the physical database details to provide for better run-time performance without affecting the application program or the control information used to describe VSAM and CA-IDMS/DB correspondences.

Note that the Mixed Page Group Binds Allowed feature may not be used with CA-IDMS/VSAM Transparency.

1.1.3 Implementation

To implement CA-IDMS/VSAM Transparency, follow these steps:

1. **Prepare the database** by following the procedures detailed in *CA-IDMS Database Administration*.
2. **Transfer data from VSAM files to the CA-IDMS/DB database** by using the CA-IDMS/VSAM Transparency migration utility. This utility is described in Chapter 5, "Runtime Operations" on page 5-1.
3. **Prepare the control tables** that will be used by CA-IDMS/VSAM Transparency at runtime. Instructions on preparing control tables are presented in Chapter 4, "Preparing Control Information" on page 4-1.
4. **Modify the application JCL.** Sample JCL is presented in Chapter 5, "Runtime Operations" on page 5-1.

1.2 General architecture

CA-IDMS/VSAM Transparency is a front-end to back-end design that contains dedicated modules and routines. The modules used by CA-IDMS/VSAM Transparency are summarized below.

- **An operating system services manager interface module** provides operating-system-dependent open and close services for CA-IDMS/VSAM Transparency files. There is one system services manager for each operating system that CA-IDMS/VSAM Transparency supports.
 - **Front-end and back-end request processing modules** perform:
 - Open and close processing that is operating system independent, such as the acquisition of buffer space and BIND requests
 - All simulation of VSAM functions, after a CA-IDMS/VSAM Transparency file is opened or closed
 - **A CICS interface module** (used in the CICS environment only) allows communication to occur between CICS transactions and the CA-IDMS/VSAM Transparency front-end module.
 - **Control tables** define the correspondence between VSAM data structures and CA-IDMS/DB data sets. There are two types of control tables in CA-IDMS/VSAM Transparency:
 - The **file management table (FMT)** maps each VSAM data structure used by the application program to a CA-IDMS/DB database record and set. This table is required for each VSAM data structure used, regardless of the processing environment (batch or CICS).
 - The **transaction name table (TNT)** maps the CICS online application transaction names to CA-IDMS/DB subschema names. This table is optional for the CICS environment and is not applicable to the batch environment.
- ▶▶ Information on preparing control table information is presented in Chapter 4, “Preparing Control Information” on page 4-1.

▶▶ For more detailed information on CA-IDMS/VSAM Transparency architecture, refer to Appendix A, “CA-IDMS/VSAM Transparency Architecture” on page A-1.

1.3 Operating requirements

CA-IDMS/VSAM Transparency requires the following system components:

- **Operating system** CA-IDMS/VSAM Transparency runs under the various releases supported for CA-IDMS. Please see the product information distributed with your installation package for specific release levels.
- **Memory requirements** CA-IDMS/VSAM Transparency requires the following components:

- The front end requires approximately 20K, distributed in the following areas:
 - For OS/390, in the Common System area (CSA)
 - For VSE/ESA, in the System GETVIS area

In addition, the front end causes the loading of the IDMS external request unit environment, which loads IDMS components that are needed for the passing of requests and data between the front end and the back end.

- The back end requires approximately 16K, distributed in the following areas:
 - For local mode, in the user's region/partition
 - For central version, in either a reentrant pool or a program pool
 - Each task will require space on the CV storage pool, or in local storage, of sufficient storage needed for processing a CA-IDMS/DB run unit.
- **Computer Associates software** The release of CA-IDMS/VSAM Transparency must be the same as the release of CA-IDMS/DB with which CA-IDMS/VSAM Transparency is running.
 - **Operating mode** CA-IDMS/VSAM Transparency runs in CA-IDMS/DB local mode and under the CA-IDMS/DB central version. Use of the central version allows concurrent use of the CA-IDMS/DB database by multiple application programs, including batch programs, CICS CA-IDMS/VSAM Transparency programs, and other CA-IDMS/DB applications.

Chapter 2. CA-IDMS/DB Schema and VSAM File Correspondences

2.1 Overview	2-3
2.2 Key-sequenced data set	2-4
2.2.1 Record and set representations	2-4
2.2.2 Location mode options	2-4
2.2.3 Example of schema definition	2-5
2.3 Entry-sequenced data set	2-6
2.3.1 Record and set representations	2-6
2.3.2 Location mode options	2-6
2.3.3 Examples of schema definition	2-7
2.4 Relative-record data set	2-8
2.4.1 Record and set representations	2-8
2.4.2 Location mode options	2-8
2.4.3 Example of schema definition	2-9
2.5 Alternate index	2-10
2.5.1 Record and set representations	2-10
2.5.2 Location mode options	2-10
2.5.3 Example of schema definition	2-11
2.6 Schema definition summary	2-12
2.6.1 Schema definitions for VSAM data structures	2-13
2.7 Area and file considerations	2-14

2.1 Overview

Relate VSAM structures to CA-IDMS/DB structures: CA-IDMS/VSAM Transparency supports VSAM data structures by establishing a one-to-one correspondence between a VSAM data structure and a CA-IDMS/DB non-SQL set. All VSAM data structure types are supported, including:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Relative-record data set (RRDS)
- Alternate index (PATH)

Before CA-IDMS/VSAM Transparency can establish the correspondence between VSAM and CA-IDMS/DB, you must represent the VSAM data structures in a CA-IDMS/DB schema. Each VSAM data structure corresponds to a record within a system-owned indexed set. The record type, location mode used to store the data, and set order of the indexed set vary for each VSAM data structure type.

About this chapter: The remainder of this chapter presents:

- The record definitions, set definitions, and location mode options that correspond to each of the data structures that CA-IDMS/VSAM Transparency supports
- General area and file considerations for defining CA-IDMS/VSAM Transparency data structures to the CA-IDMS/DB schema

Note: Special considerations apply to KSDS, ESDS, and PATH data structures that contain variable-length records. These considerations are described in Appendix C, “Variable-Length Record Considerations” on page C-1.

2.2 Key-sequenced data set

In VSAM, a key-sequenced data set (KSDS) contains records that are indexed by a prime key. When a KSDS is initially loaded, the physical sequence of the records may match the logical key sequence. After updates are made, the physical sequence and logical key sequence may or may not be in sync with each other.

The following rules apply to a VSAM KSDS:

- The prime key must be unique (duplicates are not allowed).
- The prime key must reside at a fixed displacement in the record-key field, even if the records are of variable length.

2.2.1 Record and set representations

In a CA-IDMS/DB database, a KSDS data set corresponds to a system-owned indexed set sorted on prime key. To represent a VSAM KSDS in a CA-IDMS/DB database, you must include the following definitions in the schema:

- A record type whose description corresponds to the KSDS record
- An indexed set whose owner is SYSTEM and whose member record is the record type described above
- A set order of SORTED (on the prime key)
- A duplicates option of DUPLICATES NOT ALLOWED

2.2.2 Location mode options

The location mode options for a KSDS data set are:

- **VIA (the indexed set)** stores records in physical key sequence. This option is recommended.
- **CALC (on prime key)** stores records by using a randomizing algorithm on the prime-key field. This option may be used for direct retrieval.
- **VIA (other set)** stores records in physical key sequence near their owner records. This option may be used if the majority of processing occurs through CA-IDMS/DB programs that access the data through the specified set. VIA (other set) is not recommended if access is primarily through CA-IDMS/VSAM Transparency.
- **CALC (on other field)** stores records by using a randomizing algorithm on a field other than the prime-key field. This option may be used if the majority of processing is through CA-IDMS/DB programs that access the data through the specified key field. This option is not recommended if access is primarily through CA-IDMS/VSAM Transparency.
- **DIRECT** stores records on or near a user-specified database page. This option may be used if the majority of processing occurs through CA-IDMS/DB programs

that access the data directly, by using the database key. It is not recommended if access is primarily through CA-IDMS/VSAM Transparency.

2.2.3 Example of schema definition

The DDL statements necessary to represent a KSDS department record in the CA-IDMS/DB database might be as follows:

```
ADD RECORD NAME IS DEPARTMENT
    LOCATION MODE IS VIA DEPARTMENT-NDX SET.
```

```
ADD SET NAME IS DEPARTMENT-NDX
    ORDER IS SORTED
    MODE IS INDEX
    OWNER IS SYSTEM
    MEMBER IS DEPARTMENT
        ASCENDING KEY IS DEPARTMENT-ID
        DUPLICATES NOT ALLOWED.
```

2.3 Entry-sequenced data set

In VSAM, an entry-sequenced data set (ESDS) contains records that are physically stored in the order in which they are entered. ESDS records can be deleted logically, but not physically; they can be updated only if the length of the record is not changed.

2.3.1 Record and set representations

In a CA-IDMS/DB database, a VSAM ESDS data set corresponds to a system-owned indexed set with a set order of SORTED (on db-key) or LAST. To represent a VSAM ESDS in a CA-IDMS/DB database, you must include the following definitions in the schema:

- A record type whose description corresponds to the ESDS record
- An indexed set whose owner is SYSTEM and whose member record is the record type described above
- A set order of SORTED (on db-key) or LAST

2.3.2 Location mode options

The location mode options for an ESDS are:

- **VIA (the indexed set)** stores records in physical key sequence. This option must be used when the set order is SORTED (on db-key) and is recommended when the set order is LAST.
- **VIA (other set)** stores records in physical key sequence near their owner records. This option may be used when the set order is LAST, if the majority of processing occurs through CA-IDMS/DB programs that access the data through the specified set. It is not recommended if access is primarily through CA-IDMS/VSAM Transparency.
- **CALC (on other field)** stores records by using a randomizing algorithm on a field other than the prime-key field. This option may be used when the set order is LAST, if the majority of processing occurs through CA-IDMS/DB programs that access the data through the specified key field. It is not recommended if access is primarily through CA-IDMS/VSAM Transparency.
- **DIRECT** stores records on or near a user-specified database page. This option may be used when the set order is LAST, if the majority of processing occurs through CA-IDMS/DB programs that access the data directly, by using the database key. It is not recommended if access is primarily through CA-IDMS/VSAM Transparency.

2.3.3 Examples of schema definition

Sorted set: The DDL statements necessary to represent a VSAM ESDS employee record in the CA-IDMS/DB database as a sorted set might be as follows:

```
ADD RECORD NAME IS EMPLOYEE
    LOCATION MODE IS VIA EMPLOYEE-NDX SET.

ADD SET NAME IS EMPLOYEE-NDX
    ORDER IS SORTED
    MODE IS INDEX
    OWNER IS SYSTEM
    MEMBER IS EMPLOYEE
    ASCENDING KEY IS DBKEY.
```

Unsorted set: The DDL statements necessary to represent a VSAM ESDS employee record in the CA-IDMS/DB database as an unsorted set might be as follows:

```
ADD RECORD NAME IS EMPLOYEE
    LOCATION MODE IS VIA EMPLOYEE-NDX SET.

ADD SET NAME IS EMPLOYEE-NDX
    ORDER IS LAST
    MODE IS INDEX
    OWNER IS SYSTEM
    MEMBER IS EMPLOYEE.
```

2.4 Relative-record data set

In VSAM, a relative-record data set (RRDS) is always preformatted into fixed-length slots that contain unique relative-record numbers. These numbers indicate a record's relative position within the file. The RRDS record length is always fixed and equal to the length of the RRDS slot, which may or may not contain a record. Records can be added, updated, or deleted only within the predefined slots; no new slots can be added after the RRDS is formatted. Direct access to an RRDS is by relative-record number.

2.4.1 Record and set representations

In a CA-IDMS/DB database, an RRDS data set corresponds to a system-owned indexed set with a set order of SORTED (on an added control field). To represent a VSAM RRDS in a CA-IDMS/DB database, you must include the following definitions in the schema:

- A record type whose description corresponds to the RRDS record.
- A four-byte control field added to the end of the record. This field will contain the relative-record number. It must be included in the subschema definition of the record for all applications that access the record.

Note: The control field will not be passed back to the application program.

- An indexed set whose owner is SYSTEM and whose member record is the record type described above.
- A set order of SORTED (on the CA-IDMS/VSAM Transparency control field that holds the relative-record number).
- A duplicates option of DUPLICATES NOT ALLOWED.

2.4.2 Location mode options

The location mode options for an RRDS are described below:

- **VIA (the indexed set)** stores records in control field sequence. This option is recommended.
- **CALC (on control field)** stores records by using a randomizing algorithm on the control field. This option may be used for direct retrieval.
- **VIA (other set)** stores records in physical key sequence near their owner records. This option may be used if the majority of processing occurs through CA-IDMS/DB programs that access the data through the specified set. VIA (other set) is not recommended if access is primarily through CA-IDMS/VSAM Transparency.
- **CALC (on other field)** stores records by using a randomizing algorithm on a field other than the prime-key field. This option may be used if the majority of processing is through CA-IDMS/DB programs that access the data through the specified key field. It is not recommended if access is primarily through CA-IDMS/VSAM Transparency.

- **DIRECT** stores records on or near a user-specified database page. This option may be used if the majority of processing occurs through CA-IDMS/DB programs that access the data directly, by using the database key. It is not recommended if access is primarily through CA-IDMS/VSAM Transparency.

2.4.3 Example of schema definition

The DDL statements necessary to represent a VSAM RRDS department record in the CA-IDMS/DB database might be as follows:

```
ADD RECORD NAME IS DEPARTMENT
    LOCATION MODE IS VIA DEPARTMENT-NDX SET.
```

```
ADD SET NAME IS DEPARTMENT-NDX
    ORDER IS SORTED
    MODE IS INDEX
    OWNER IS SYSTEM
    MEMBER IS DEPARTMENT
        ASCENDING KEY IS CONTROL-FIELD
        DUPLICATES NOT ALLOWED.
```

2.5 Alternate index

In VSAM, an alternate index is a structure that allows you to access data in an existing KSDS or ESDS data set by using an alternate symbolic key. Alternate keys may be nonunique; the alternate key and prime key can overlap. The combination of an alternate index and its base KSDS or ESDS data set is known as a **PATH**.

A VSAM alternate index may or may not be a member of the UPGRADE set for the base KSDS or ESDS. VSAM automatically maintains the index if it is a member of this set.

2.5.1 Record and set representations

In a CA-IDMS/DB database, an alternate index corresponds to a system-owned indexed set sorted on the alternate key. Index maintenance is performed automatically for all CA-IDMS/VSAM Transparency alternate indexes. The VSAM UPGRADE set is not applicable to CA-IDMS/VSAM Transparency.

To represent a VSAM alternate index in the CA-IDMS/DB database, you must include the following definitions in the schema:

- The record type whose description corresponds to the base KSDS or ESDS. Note that the base KSDS or ESDS must already be defined in the schema.
- An indexed set whose owner is SYSTEM and whose member record represents the base KSDS or ESDS.
- A set order of SORTED (on the alternate key).
- For nonunique alternate indexes, a duplicates option of LAST.
- For unique alternate indexes, a duplicates option of DUPLICATES NOT ALLOWED.

2.5.2 Location mode options

A record can have only one location mode in CA-IDMS/DB. You can choose one of the location mode options described earlier in this section for the base KSDS or ESDS record. Or, if an alternate index exists, you can choose one of the following location mode options for the base record:

- **VIA (the alternate index set)** stores records in physical alternate key sequence. This option may be used for sequential processing through the alternate index.
- **CALC (on alternate key)** stores records by using a randomizing algorithm on the alternate key. This option may be used for direct retrieval through the alternate key.

2.5.3 Example of schema definition

Sample DDL statements that represent a VSAM KSDS with an alternate index in the CA-IDMS/DB database are shown below. In this example, the DEPARTMENT record is stored via the alternate index set.

Primary index:

```
ADD RECORD NAME IS DEPARTMENT
    LOCATION MODE IS VIA DEPARTMENT-NDX SET.

ADD SET NAME IS DEPARTMENT-NDX      <---- Primary index
    ORDER IS SORTED
    MODE IS INDEX
    OWNER IS SYSTEM
    MEMBER IS DEPARTMENT
        ASCENDING KEY IS CONTROL-FIELD
        DUPLICATES NOT ALLOWED.
```

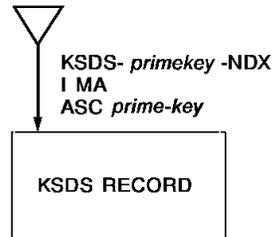
Alternate index:

```
ADD SET NAME IS DEPARTMENT-NAME-NDX  <---- Alternate index
    ORDER IS SORTED
    MODE IS INDEX
    OWNER IS SYSTEM
    MEMBER IS DEPARTMENT
        ASCENDING KEY IS DEPARTMENT-NAME
        DUPLICATES LAST.
```

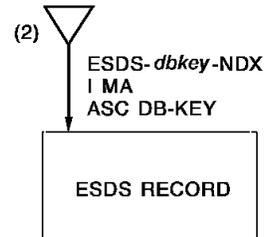
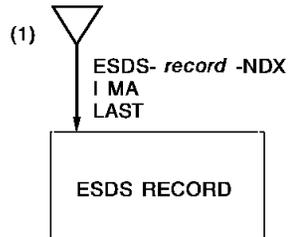
2.6 Schema definition summary

The following diagram illustrates the set orders that establish correspondences between CA-IDMS/DB and VSAM data structures. VSAM data structures correspond to system-owned indexed sets.

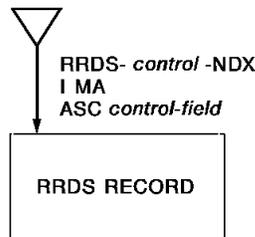
KSDS



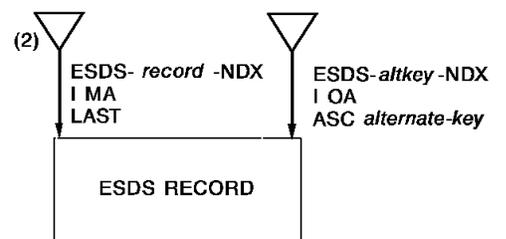
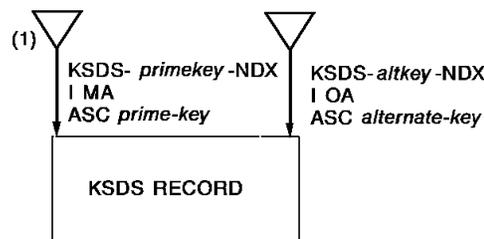
ESDS



RRDS with control field



Alternate index



2.6.1 Schema definitions for VSAM data structures

The following table shows the CA-IDMS/DB schema definitions for each VSAM data structure type.

CA-IDMS/DB schema defi- nitions	KSDS data structure	ESDS data structure	RRDS data structure	Alternate index data structure
Record type and set member	KSDS record definition	ESDS record definition	RRDS record definition	
Set type	System-owned index	System-owned index	System-owned index	System- owned index
Set order	Sorted on prime key	Sorted on db-key or LAST	Sorted on the added control field	Sorted on the alternate key
Duplicates	Not allowed	Not allowed	Not allowed	Not allowed or LAST
Location mode	VIA indexed set or other set CALC prime key or other field DIRECT	VIA indexed set or other set CALC other field DIRECT	VIA indexed set or other set CALC control field or other field DIRECT	VIA alter- nate key CALC alternate key

2.7 Area and file considerations

To represent a VSAM data structure in the CA-IDMS/DB database, you must specify the area in which you want the data to reside. Additionally, you must associate the area with a CA-IDMS/DB file, and assign the file to an external ddname (OS/390) or filename (VSE/ESA).

►► Instructions for defining areas and files are presented in *CA-IDMS Database Administration*.

Design considerations: The following design considerations apply to schema area and file definitions for VSAM data structures:

- You can put any number of CA-IDMS/VSAM Transparency files in the same CA-IDMS/DB area, as long as you have space for these files. If you put more than one CA-IDMS/VSAM Transparency file in the same area, each file will be readied in the same usage mode.

Note: For optimal performance, you should put only one CA-IDMS/VSAM Transparency file in an area.

- Each ddname (OS/390) or filename (VSE/ESA) must be unique for an application program. This means that the external file name specified in the ASSIGN TO clause of the physical database DDL FILE statement must be different than the file name used in the VSAM application.

For example, suppose a VSAM application program contains the following COBOL SELECT statement:

```
SELECT CUSTFILE ASSIGN TO SYS030.
```

You could code the following file assignment:

```
ADD FILE CUSTOMER-FILE ASSIGN TO CSTFILE.
```

The JCL for running the VSAM application in local mode will include one DD or DLBL statement for each of the files described above:

```
//SYS030 DD SUBSYS=(ESVS,'FMT=fmtname','SUBSCHEMA=ssname',... )  
//CSTFILE DD DSN=CUST-FILE,DISP=SHR
```

►► For more information on CA-IDMS/DB database definition, refer to *CA-IDMS Database Administration*.

Chapter 3. Record Access and Processing

- 3.1 Overview 3-3
- 3.2 Keyed access 3-4
 - 3.2.1 Keyed direct processing 3-4
 - 3.2.2 Keyed sequential processing 3-4
 - 3.2.3 Keyed skip-sequential processing 3-5
 - 3.2.4 Keyed backward processing 3-5
- 3.3 Addressed access 3-6
 - 3.3.1 Addressed direct processing 3-6
 - 3.3.2 Addressed sequential processing 3-6
 - 3.3.3 Addressed backward processing 3-6
- 3.4 DML correspondences to VSAM commands 3-7
- 3.5 Additional VSAM Options 3-8

3.1 Overview

Processing options: Both VSAM and CA-IDMS/VSAM Transparency allow data access by key or by address. VSAM can access and process records by relative byte displacement from the beginning of the file (RBA), by relative-record number (RRN), or by prime or alternate key. CA-IDMS/DB accesses and processes records through Data Manipulation Language (DML) statements that operate on schema-defined records and sets.

At runtime, CA-IDMS/VSAM Transparency translates the most commonly used VSAM processing options into DML statements. This translation is transparent to the VSAM program.

About this chapter: The remainder of this chapter discusses the VSAM processing modes and their DML correspondences presented by:

- Keyed access method
- Addressed access method

These discussions are followed by a table that summarizes the correspondences between VSAM commands and DML commands, and a discussion of additional VSAM options.

3.2 Keyed access

CA-IDMS/VSAM Transparency supports keyed access for key-sequenced data sets, relative-record data sets, and alternate index paths. VSAM uses prime keys, alternate keys, or relative-record numbers to locate records; CA-IDMS/VSAM Transparency translates the VSAM keys into sort keys for system-owned indexed sets.

Keyed access processing methods: Keyed access can be used with:

- Direct processing
- Sequential processing
- Skip-sequential processing
- Backward processing

Direct, sequential, skip-sequential and backward processing are discussed below.

3.2.1 Keyed direct processing

Processing method: VSAM keyed direct processing locates records by using a GET call with a user-supplied search argument. The search condition can be either equal to a whole key, or greater than or equal to the leading portion of a key.

Keyed direct processing does not depend on any previously established position within the VSAM file; however, it can set the position for subsequent requests.

DML commands CA-IDMS/VSAM Transparency translates VSAM calls for direct GETs into the OBTAIN WITHIN SET USING SORTKEY DML command.

3.2.2 Keyed sequential processing

Processing method: VSAM keyed sequential processing works as follows:

- A position in the file can be established with a POINT call that uses a prime key, alternate key, or relative-record number.
- Records are then retrieved in ascending, logical, keyed sequential order (GET NEXT).

DML commands: CA-IDMS/VSAM Transparency translates VSAM calls for keyed sequential processing into the following commands:

- A POINT call is translated into the FIND WITHIN SET USING SORTKEY DML command.
- A GET call is translated into the OBTAIN FIRST/NEXT WITHIN SET DML command.

3.2.3 Keyed skip-sequential processing

Processing method: VSAM keyed skip-sequential (SKP-sequential) processing works in a way similar to VSAM keyed sequential processing:

- A position in the file can be established with a POINT call that uses a prime key, alternate key, or a relative-record number.
- Records are then retrieved in ascending, logical, keyed sequential order (GET NEXT).

User-specified skips to other positions in the file can be made to a key or relative-record number that is greater than the current key or relative-record number. Backward SKP-sequential processing is not allowed.

DML commands: CA-IDMS/VSAM Transparency translates VSAM calls for keyed SKP-sequential processing into the following commands:

- A POINT call is translated into the FIND WITHIN SET USING SORTKEY DML command.
- A GET call is translated into the OBTAIN FIRST/NEXT WITHIN SET DML command.

3.2.4 Keyed backward processing

Processing method: VSAM keyed backward processing accesses records sequentially in descending order from a specified position in the file. The starting position in the file can be established with a POINT call either on a prime key or on a relative-record number. Backward processing cannot be used with SKP-sequential processing.

DML commands: CA-IDMS/VSAM Transparency translates VSAM calls for keyed backward processing into the following commands:

- A POINT call is translated into the FIND WITHIN SET USING SORTKEY DML command.
- A GET call is translated into the OBTAIN LAST/PRIOR WITHIN SET DML command.

3.3 Addressed access

CA-IDMS/VSAM Transparency supports addressed access for key-sequenced data sets and entry-sequenced data sets. When addressed access is used, CA-IDMS/VSAM Transparency uses the CA-IDMS/DB db-key in place of the RBA.

Addressed access can be used with:

- Direct processing
- Sequential processing
- Backward processing

Direct, sequential, and backward processing are discussed below.

3.3.1 Addressed direct processing

Processing method: VSAM addressed direct processing allows access to a record through specification of the RBA. No generic partial key is allowed. Although direct processing is independent of any previously established position within the file, it can set the position for subsequent requests.

DML commands: CA-IDMS/VSAM Transparency translates VSAM calls for addressed direct processing into the OBTAIN DBKEY DML command.

3.3.2 Addressed sequential processing

Processing method: VSAM addressed sequential processing allows access to records in physical order by RBA. Access can be either from the beginning of the file or from an established position in the file. Specification of the RBA in a call is not allowed.

DML commands: CA-IDMS/VSAM Transparency translates VSAM calls for addressed sequential processing into the OBTAIN FIRST/NEXT WITHIN SET DML command.

3.3.3 Addressed backward processing

Processing method: VSAM addressed backward processing accesses records in descending order from a position in the file that can be established with a POINT call on an RBA. Backward processing cannot be used with SKP-sequential processing.

DML commands: CA-IDMS/VSAM Transparency translates VSAM calls for addressed backward processing into the OBTAIN LAST/PRIOR WITHIN SET DML command.

3.4 DML correspondences to VSAM commands

CA-IDMS/VSAM Transparency converts VSAM commands to DML commands, which are passed to the database management system for processing.

The following table shows the DML correspondences to VSAM commands.

VSAM Command	CA-IDMS/DB DML Command
READ BY KEY	OBTAIN WITHIN SET USING SORTKEY
START/STARTBR	FIND WITHIN SET USING SORTKEY
READ NEXT	OBTAIN FIRST/NEXT WITHIN SET
READ PREVIOUS	OBTAIN LAST/PRIOR WITHIN SET
WRITE	STORE
REWRITE	MODIFY
DELETE	ERASE

3.5 Additional VSAM Options

Supported options: In addition to the processing modes described above, CA-IDMS/VSAM Transparency supports these options:

- Chained request parameter lists
- Concurrent access
- Reusable files
- Variable-length records
- The following OS/390 VSAM exits:
 - End of data (EODAD) exit
 - Logical error (LERAD) exit
 - Physical error (SYNAD) exit
 - Journal (JRNAD) exit, which will be ignored by CA-IDMS/VSAM Transparency
 - User wait (UPAD) exit, which will be ignored by CA-IDMS/VSAM Transparency
- The following VSE/ESA VSAM exits:
 - End of data (EODAD) exit
 - Logical error (LERAD) exit
 - Physical error (SYNAD) exit
 - Journal (JRNAD) exit, which will be ignored by CA-IDMS/VSAM Transparency
 - User wait (EXCPAD) exit, which will be ignored by CA-IDMS/VSAM Transparency
- User exits
- Synchronous processing
- Asynchronous processing

Unsupported options: These less frequently used processing options are not supported by CA-IDMS/VSAM Transparency:

- Index-component-processing user buffering
- User-buffering deferred writes and transaction ids
- Deferred writes and transaction ids
- Shared resources
- Control interval access
- ISAM interface

- VSAM error message area
- The exception exit
- The user security verification exit

Chapter 4. Preparing Control Information

- 4.1 Overview 4-3
- 4.2 Compiler-directive statements 4-5
 - 4.2.1 Example 4-6
- 4.3 File management table 4-7
 - 4.3.1 Usage 4-10
 - 4.3.2 Examples 4-12
 - 4.3.3 FMT JCL 4-13
- 4.4 Transaction name table 4-17
 - 4.4.1 Examples 4-18
 - 4.4.2 TNT JCL 4-19

4.1 Overview

FMT and TNT tables: CA-IDMS/VSAM Transparency uses two control tables to establish the correspondences between VSAM and the database:

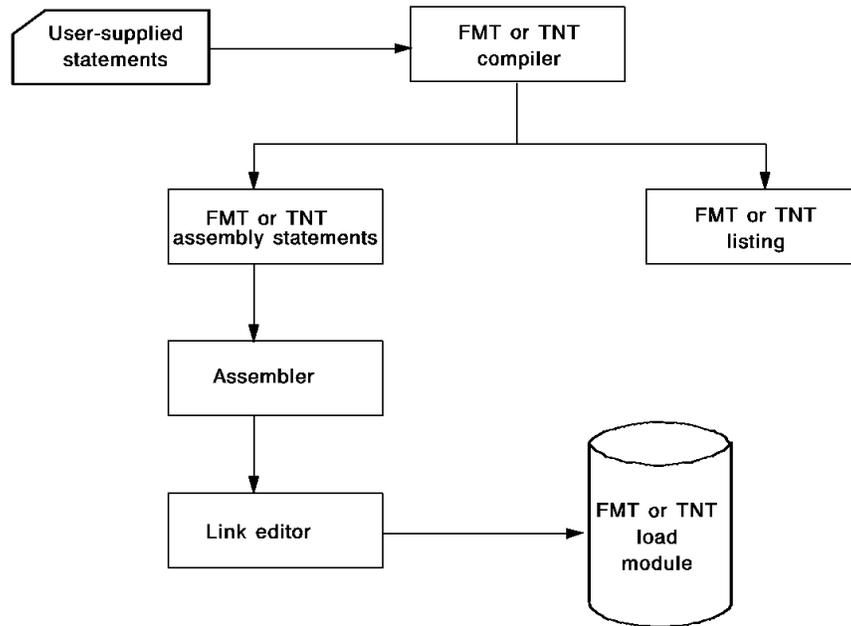
- The **file management table (FMT)** maps VSAM data structures to CA-IDMS/DB records and sets. CA-IDMS/VSAM Transparency requires one file management table for each VSAM data set to be processed.
- The **transaction name table (TNT)** maps CICS transactions to CA-IDMS/DB subschemas. The TNT is optional if all transactions are to use the same subschema; it is required if different subschemas are to be used.

Before an application program can be run with CA-IDMS/VSAM Transparency, you must compile, assemble, and link edit the appropriate FMTs and TNTs. Only one FMT or TNT can be compiled in a single execution of the corresponding compiler.

About this chapter: This chapter describes how to execute file management and transaction name tables and presents information as follows:

- **Compiler-directive statements** specify in-stream processor control information for the FMT and TNT compilers.
- **FMT statements** establish the correspondence between VSAM data structures and CA-IDMS/DB records and sets.
- **TNT statements** establish the correspondence between CICS transaction names and CA-IDMS/DB subschema names.

Flow diagram: The following diagram illustrates the compilation, assembly, and link editing of the user-supplied FMT or TNT statements that must occur before an application program can be run with CA-IDMS/VSAM Transparency.

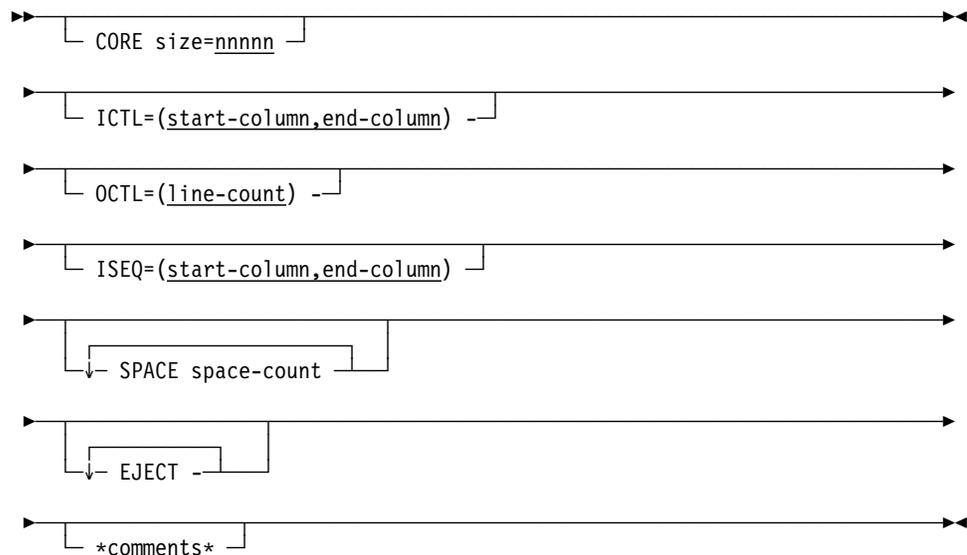


4.2 Compiler-directive statements

Purpose: Compiler-directive statements specify:

- The amount of storage required to compile a control table
- The range of input columns within which control table statements can be coded
- The sequence checking of input to the compiler
- The formatting of compiler report output

Syntax



Parameters

CORE size=nnnnnn

Specifies the amount of storage the compiler is to acquire (by a GETMAIN under OS/390 or by a GETVIS or COMREG under VSE/ESA) for the VSAM CA-IDMS/VSAM Transparency control table being generated. *Nnnnnn* is a 1- to 6-digit numeric value.

The amount of storage acquired is rounded up to the next doubleword by the compiler. If CORE SIZE is not coded, the compiler automatically acquires 48K of storage.

ICTL=(start-column,end-column)

Identifies the columns within which control table input statements can be coded. These columns must be in the range 1 through 80. Default values are 1 and 80, respectively. If coded, this statement must precede input for the CA-IDMS/VSAM Transparency control table.

OCTL=(line-count)

Specifies the number of printed lines per page of printed output. The value specified must be a number in the range 1 through 66. The default value is 60. If

coded, this statement must precede input for the CA-IDMS/VSAM Transparency control table.

ISEQ=(start-column,end-column)

Causes the compiler to check the sequencing of all input. The start and end columns of the sequence number generated for each input statement have numeric values that range from 1 through 80. The maximum allowable difference between entries is 10. If coded, this statement must precede input for the CA-IDMS/VSAM Transparency control table.

SPACE *space-count*

Causes the compiler to skip the specified number of lines on the output report. *Space-count* is a number in the range 1 through 9. One blank is allowed between SPACE and *space-count*. Several SPACE statements can appear in the compiler input.

EJECT

Directs the compiler to stop printing the current page and to begin printing a new page. This statement must occupy a line by itself and can be inserted between control table input statements.

comments

Directs the compiler to interpret as comments those characters placed after an initial asterisk. Comments can be embedded in control table statements; they are terminated automatically at the end of the input line, unless the compiler encounters a second asterisk in the line. A second asterisk causes explicit termination.

Note: Be sure to use an even number of asterisks when you are writing comments. An odd number of asterisks will turn the comments back on.

4.2.1 Example

The following example illustrates entries for the compiler-directive statements:

```
CORE=16
ICTL=(1,72)
OCTL=(45)
ISEQ=(3,72)
SPACE 2
EJECT
* END OF STATEMENTS *
```

4.3 File management table

Purpose: The file management table is a control block that establishes a one-to-one correspondence between the VSAM data set referenced by the application program and the CA-IDMS/DB database record accessed by CA-IDMS/VSAM Transparency. Each CA-IDMS/VSAM Transparency file is required to have a file management table. User-supplied parameters are compiled with the FMT compiler and linked into the CA-IDMS/DB load library or core-image library.

You must define and compile one FMT for each CA-IDMS/VSAM Transparency data set used by the VSAM application programs. FMT control statements must be entered in the order presented in the syntax below.

►► See the tables located under 4.3.1, “Usage” on page 4-10 later in this section for VSAM data set definitions, CA-IDMS schema definitions and corresponding FMT compiler statements. See Appendix G, “VSE/ESA CICS SYSESVS Parameters” on page G-1 later in this manual for VSE/ESA CICS SYSESVS file management parameters.

Syntax

```
►► FMT NAME is file-management-table-name —.—————►
► FILE TYPE is —┬─ KSDS —————┬─.►
                  └─ ESDS —————┬─
                  └─ RRDS —————┬─
                  └─ PATH —————┬─
                                └─ BASE —┬─ cluster ─┬─ is ┬─ KSDS ← ┬─
                                      └─ ESDS ─┬─
```

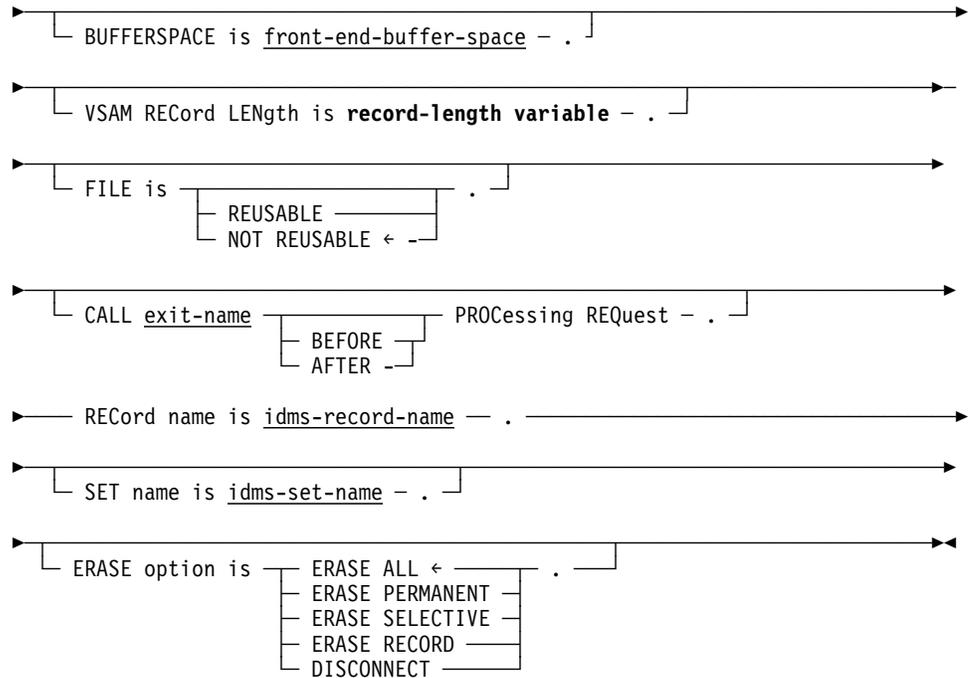
Required for KSDS and a PATH only

```
►► KEY LENGTH is vsam-key-length —.—————►
```

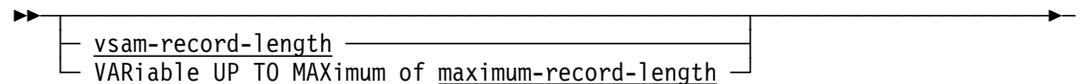
```
► KEY POSITION is vsam-key-displacement —.—————►◀
```

Required and valid for RRDS only

```
►► RELative RECOrd NUmber POSition is control-field-displacement —.——►◀
```



Expansion of record-length variable:



Parameters: You must enter FMT control statements in the order presented below.

FMT NAME is file-management-table-name

Specifies a 1- to 8-character user-defined FMT name. FMT NAME is a required statement.

FILE TYPE is KSDS/ESDS/RRDS/PATH

Identifies the type of VSAM file that you are defining to CA-IDMS/VSAM Transparency. FILE TYPE is a required statement.

▶▶ See 4.3.1, “Usage” on page 4-10 for specific VSAM cluster statements that indicate the file type.

KEY LENGth is vsam-key-length

Specifies the length of the key of a KSDS record or a PATH. *Vsam-key-length* can be a 1- to 3-digit value in the range 1 through 255. Note that the KEY LENGTH statement is required for a KSDS and a PATH, and is invalid for a RRDS and an ESDS.

KEY POSition is vsam-key-displacement

Specifies the location of the key in a KSDS or PATH record. *Vsam-key-displacement* can be 1 to 5 digits in the range 0 through 32,767. (The first byte of

a record is considered to be position zero.) This statement is required for a KSDS and a PATH; it is invalid for a RRDS and an ESDS.

BUFFERSPACE is front-end-buffer-space

Specifies the amount of buffer space to be reserved by the front-end module to receive the results of VSAM LOCATE MODE requests. Buffer space can be allocated for any of the data set types. If not specified, this variable defaults to the VSAM record length.

Front-end-buffer-space can be a 1- to 5-digit value that ranges from the maximum VSAM record length to 32,767.

CA-IDMS/VSAM Transparency automatically reserves buffer space by using the greatest of the following values:

- The buffer space value specified in the access method control block (ACB)
- The greater of the following two values:
 - ((String number or number of data buffers specified in the ACB) * maximum VSAM record length)
 - The buffer space value specified in the FMT

VSAM RECOrd LENgth is vsam-record-length

Specifies the length of the VSAM record for any of the VSAM data set types. The length specified can be different from that of the CA-IDMS/DB record. For a RRDS, the length specified should not include the control-field length.

VARIABLE UP TO MAXimum of maximum-record-length

Optional. Specifies the size of the largest possible record for a variable-length record. The numeric value can be 1 to 5 digits in the range 1 through 32,768.

FILE is REUSABLE/NOT REUSABLE

Specifies whether the file can be reset when opened. When a reusable file is opened RESET, all existing records are erased. The default is NOT REUSABLE. FILE IS REUSABLE/NOT REUSABLE does not apply to a RRDS.

CALL exit-name **BEFORE/AFTER PROCessing REQuest**

Names the exit routine that will be invoked either before or after DML calls are issued by CA-IDMS/VSAM Transparency. *Exit-name* can be 1 to 8 characters in length. Note that duplicate names are allowed within a single FMT or in different FMTs.

RELative RECOrd NUMBER POSition is control-field-displacement

Specifies the position within an CA-IDMS/DB record of a user-defined relative-record-number control field for an RRDS record. *Control-field-displacement* can be 1 to 5 digits in the range 0 through 32,764. (The first byte of a record is considered to be position zero.) RELATIVE RECORD NUMBER POSITION is a required statement for RRDS data sets and is invalid for KSDS, ESDS, and PATH data sets.

The user-assigned relative-record number constitutes an RRN control field that requires four additional bytes in the schema record definition. The control field must be placed at the end of the CA-IDMS/DB record.

RECORD name is idms-record-name

Names the CA-IDMS/DB record that corresponds to a VSAM record of each data set type used. RECORD NAME can be 1 to 16 characters in length.

SET name is idms-set-name

Names the CA-IDMS/DB system-owned indexed set. SET NAME is required for all file types, with the exception of RRDS data sets. *Idms-set-name* can be 1 to 16 characters in length.

**ERASE option is ERASE ALL/ERASE PERMANENT/ERASE
SELECTIVE/ERASE RECORD/DISCONNECT**

Identifies the type of ERASE that CA-IDMS/VSAM Transparency will perform when processing VSAM ERASE requests. The default is ERASE ALL.

4.3.1 Usage

Input preparation: Parameters for the user-supplied input to the file management table are taken from two sources:

- The VSAM data set definition (from LISTCAT)
- The schema record description

The table below presents the correspondences between VSAM data set definitions and FMT compiler statements.

Data set attribute	VSAM data set definition	FMT compiler statement
KSDS	DEFINE CLUSTER INDEXED (IXD)	FILE TYPE IS KSDS
ESDS	DEFINE CLUSTER NONINDEXED (NIXD)	FILE TYPE IS ESDS
RRDS	DEFINE CLUSTER NUMBERED (NUMD)	FILE TYPE IS RRDS.
PATH	DEFINE PATH PATHENTRY (alternate index or base cluster)	FILE TYPE IS PATH
Key length	DEFINE CLUSTER KEYS (length, displacement)	KEY LEN IS length
Key position	DEFINE CLUSTER KEYS (length, displacement)	KEY POS IS displacement
Buffer space	DEFINE CLUSTER BUFSPC (size)	BUFFERSPACE IS size
Record size (fixed-length record)	DEFINE CLUSTER RECSZ (min length, max length)	VSAM REC LEN IS max length
Record size (variable-length record)	DEFINE CLUSTER RECSZ (min length, max-length)	VSAM REC LEN IS VAR UP TO MAX max-length
Relative-record-number position	Not defined	REL REC NUM POS IS position
Reusable file	REUSE	FILE IS REUSABLE
Nonreusable file	NOREUSE	FILE IS NOT REUSABLE

The following table provides the correspondences between CA-IDMS/DB schema definitions and FMT compiler statements.

Attribute	CA-IDMS/DB schema definition	Sample FMT statement
Record name	ADD RECORD NAME IS DEPARTMENT.	REC IS DEPARTMENT.
Set name	ADD SET NAME IS DEPARTMENT-NDX	SET IS DEPARTMENT-NDX

4.3.2 Examples

The following tables show the required FMT statements used for VSAM KSDS, VSAM ESDS, VSAM RRDS, and VSAM PATH.

VSAM KSDS: In a sample employee database, the required FMT statements for a VSAM KSDS and the corresponding LISTCAT description might be as follows:

FMT statements	LISTCAT description
FMT NAME IS EMPFMT	
FILE TYPE IS KSDS	INDEXED
KEY LENGTH IS 44	KEYLEN-----44
KEY POSITION IS 0	RKP-----0
VSAM RECORD LENGTH IS 805	MAXLRECL-----805
FILE IS NOT REUSABLE	NOREUSE
RECORD NAME IS EMPLOYEE	
SET NAME IS EMP-NDX	

VSAM ESDS: The required FMT statements for a VSAM ESDS and the corresponding LISTCAT description might be as follows:

FMT statements	LISTCAT description
FMT NAME IS DEPFMT	
FILE TYPE IS ESDS	NONINDEXED
VSAM RECORD LENGTH IS 17	MAXLRECL-----17
FILE IS REUSABLE	REUSE
RECORD NAME IS DEPARTMENT	
SET NAME IS DEPT-NDX	

VSAM RRDS: The required FMT statements for a VSAM RRDS and the corresponding LISTCAT description might be as follows:

FMT statements	LISTCAT description
FMT NAME IS EXPFMT	
FILE TYPE IS RRDS	NUMBERED
RELATIVE RECORD NUMBER POSITION IS 9	
VSAM RECORD LENGTH IS 9	MAXLRECL-----9
RECORD NAME IS EXPERTISE	
SET NAME IS EXP-NDX	

VSAM PATH: The required FMT statements for a **VSAM PATH** and the corresponding LISTCAT description might be as follows:

FMT statements	LISTCAT description
FMT NAME IS MGT FMT	
FILE TYPE IS PATH	
BASE CLUSTER IS KSDS	
KEY LENGTH IS 8	KEYLEN-----8
KEY POSITION IS 5	RKP-----5
VSAM RECORD LENGTH IS 17	MAXLRECL-----17
FILE IS REUSABLE	REUSE
RECORD NAME IS EMPLOYEE	
SET NAME IS MANAGES-NDX	

4.3.3 FMT JCL

The JCL used to compile, assemble, and link edit an FMT is presented below. **FMT (OS/390)**

4.3 File management table

```
//*****  
//*  
//*      COMPILE FMT      *  
//*  
//*****  
//FMTC      EXEC PGM=ESVSFMT,REGION=512K  
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR  
           DD DSN=idms.loadlib,DISP=SHR  
//SYSLST DD SYSOUT=A,DCB=BLKSIZE=133  
//SYSPCH DD DSN=&&fmt,DISP=(NEW,PASS)  
           DCB=(RECFM=FB,LRECL=80,BLKSIZE=4560),  
           SPACE=(CYL,(1,1)),  
           UNIT=disk  
//SYSIDMS DD *  
Input SYSIDMS parameters, as required  
//SYSIPT DD *  
Insert FMT input statements here  
/*  
//*****  
//*  
//*      ASSEMBLE COMPILER OUTPUT      *  
//*  
//*****  
//ASM      EXEC PGM=ASMA90,PARM='NOLOAD,DECK',COND=(4,LT,S1),  
           REGION=512K  
//  
//SYSPRINT DD DUMMY  
//SYSUT1 DD UNIT=disk,SPACE=(CYL,(2,2))  
//SYSUT2 DD UNIT=disk,SPACE=(CYL,(2,2))  
//SYSUT3 DD UNIT=disk,SPACE=(CYL,(2,2))  
//SYSPUNCH DD DSN=&&fmtobj,UNIT=disk,DISP=(NEW,PASS),  
           SPACE=(80,(400,40))  
//SYSIN DD DSN=&&fmt,DISP=(OLD,PASS)  
//*****  
//*  
//*      LINK FMT TO CA-IDMS/DB LOAD LIBRARY      *  
//*  
//*****  
//LINK     EXEC PGM=IEWL,PARM='XREF,LIST,LET,SIZE=(196K,12K)',  
           REGION=512K  
//  
//SYSPRINT DD SYSOUT=A  
//SYSLIN DD DSN=&&fmtobj,DISP=(OLD,DELETE)  
//SYSUT1 DD UNIT=disk,SPACE=(TRK,(20,5))  
//SYSLMOD DD DSN=user.loadlib(fmtname),DISP=SHR
```

<code>idms.dba.loadlib</code>	Data set name of the CA-IDMS/DB load library containing CA-IDMS DMCL and database name table load modules
<code>idms.loadlib</code>	Data set name of the CA-IDMS/DB load library containing ESVSFMTC
<code>&&fmt</code>	Temporary data set that contains output from the compile step
<code>disk</code>	Symbolic device type for the disk file
<code>&&fmtobj</code>	Temporary data set that contains output from the assembly step
<code>user.loadlib</code>	Data set name of the user load library where the FMT load module will be placed
<code>fmtname</code>	Name of the FMT load module
<code>SYSIDMS</code>	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

FMT (VSE/ESA)

```

*****
*
*          COMPILE FMT
*
*****
// EXEC PROC=IDMSLBLS
// DLBL   userc1,'user.userc1'
// EXTENT ,nnnnn
// LIBDEF CL,T0=userc1,FROM=cdmsc1,SEARCH=ciln
// DLBL   IDMSPC1,'work',,SD
// EXTENT SYS020,nnnnn,,,ssss,1111
// ASSGN  SYS020,disk
// EXEC   ESVSFMTC
Input SYSIDMS parameters, as required
Insert FMT input statements here
/*
*****
*
*          ASSEMBLE COMPILER OUTPUT
*
*****
// DLBL   IJSYSIN,'work',,SD
// EXTENT SYSIPT,nnnnn,,,ssss,1111
// ASSGN  SYSIPT,disk
// OPTION CATAL
// PHASE  fmtname,*
// EXEC  ASMA90
// CLOSE SYSIPT,SYSRDR
/*
*****
*
*          LINK FMT TO CA-IDMS/DB CORE-IMAGE LIBRARY
*
*****
// EXEC LNKEDT
/&

```

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, database, SYSIDMS, and other CA-IDMS files. ►►For a complete listing of IDMSLBLS, see Appendix H, “IDMSLBLS Procedure” on page H-1.
nnnnn	Volume serial number of disk unit
userc1	Filename of a user core-image library
cdmsc1	Filename of the CA-IDMS/DB core-image library
user.userc1	File-id of a user core-image library
work	File-id of work file
ssss	Relative starting track of disk file
1111	Number of tracks required for disk file
disk	Device assignment of disk file
fmtname	Name of the FMT PHASE

DBNAME is database-name

Specifies the name of the database to be accessed by CA-IDMS/VSAM Transparency for this transaction. *Database-name* can be from 1 to 8 characters long.

4.4.1 Examples

Example 1: In the following example, all CA-IDMS/VSAM Transparency CICS transactions use the EMPSS01 subschema. Because this subschema is specified in the application JCL, no TNT has been defined.

Example 2: In this example, the EMP1 transaction uses the EMPSS01 subschema. All other transactions use the subschema specified in the application JCL, or from the SYSESVS parameters (under VSE/ESA CICS only).

▶▶See Appendix G, “VSE/ESA CICS SYSESVS Parameters” on page G-1 for more information about VSE/ESA CICS SYSESVS parameters.

```
TNT NAME IS EMPTNT.  
  
TRANSACTION NAME IS EMP1  
SUBSCHEMA NAME IS EMPSS01.
```

Example 3: In this example, the EMP1 and EMP2 transactions use the EMPSS01 subschema, the EMP3 transaction uses the EMPSS02 subschema, the ORD1 transaction uses the ORDSS01 subschema, and the ORD2 transaction uses the ORDSS02 subschema. All other transactions use the subschema specified in the application JCL, or from the SYSESVS parameters (under VSE/ESA CICS only).

▶▶See Appendix G, “VSE/ESA CICS SYSESVS Parameters” on page G-1 for more information about VSE/ESA CICS SYSESVS parameters.

```
TNT NAME IS EMPTNT.  
  
TRANSACTION NAME IS EMP1  
SUBSCHEMA NAME IS EMPSS01.  
  
TRANSACTION NAME IS EMP2  
SUBSCHEMA NAME IS EMPSS01.  
  
TRANSACTION NAME IS EMP3  
SUBSCHEMA NAME IS EMPSS02.  
  
TRANSACTION NAME IS ORD1  
SUBSCHEMA NAME IS ORDSS01.  
  
TRANSACTION NAME IS ORD2  
SUBSCHEMA NAME IS ORDSS02.
```

4.4.2 TNT JCL

The JCL used to compile, assemble, and link edit a TNT is presented below. **OS/390 (TNT)**

```
//*****
//*
//*      COMPILE TNT
//*
//*****
//TNTC    EXEC PGM=ESVSTNTC,REGION=512K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//        DD DSN=idms.loadlib,DISP=SHR
//SYSLST  DD SYSOUT=A,DCB=BLKSIZE=133
//SYSPCH  DD DSN=&&tnt,DISP=(NEW,PASS)
//        DCB=(RECFM=FB,LRECL=80,BLKSIZE=4560),
//        SPACE=(CYL,(1,1)),
//        UNIT=disk
//SYSIDMS DD *
Input SYSIDMS parameters, as required
//SYSIPT  DD *
Insert TNT input statements here
/*
//*****
//*
//*      ASSEMBLE COMPILER OUTPUT
//*
//*****
//ASM     EXEC PGM=ASMA90,PARM='NOLOAD,DECK',COND=(4,LT,S1),
//        REGION=512K
//SYSPRINT DD DUMMY
//SYSUT1  DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSUT2  DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSUT3  DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSPUNCH DD DSN=&&tntobj,UNIT=disk,DISP=(NEW,PASS),
//        SPACE=(80,(400,40))
//SYSIN   DD DSN=&&tnt,DISP=(OLD,DELETE)
//*****
//*
//*      LINK TNT TO CA-IDMS/DB LOAD LIBRARY
//*
//*****
//LINK    EXEC PGM=IEWL,PARM='XREF,LIST,LET,SIZE=(196K,12K)',
//        REGION=512K
//SYSPRINT DD SYSOUT=A
//SYSLIN  DD DSN=&&tntobj,DISP=(OLD,DELETE)
//SYSUT1  DD UNIT=disk,SPACE=(TRK,(20,5))
//SYSLMOD DD DSN=user.loadlib(tntname),DISP=SHR
```

4.4 Transaction name table

<code>idms.dba.loadlib</code>	Data set name of the CA-IDMS/DB load library containing CA-IDMS DMCL and database name table load modules
<code>idms.loadlib</code>	Data set name of the CA-IDMS/DB load library containing ESVSTNTC
<code>&&tnt</code>	Temporary data set that contains output from the compile step
<code>disk</code>	Symbolic device type for disk file
<code>&&tntobj</code>	Temporary data set that contains output from the assembly step
<code>user.loadlib</code>	Data set name of the user load library where the TNT load module will be placed
<code>tntname</code>	Name of the TNT load module
<code>SYSIDMS</code>	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

VSE/ESA (TNT)

```
*****
*
*          COMPILE TNT
*
*****
//EXEC PROC=IDMSLBLS
//DLBL   userc1,'user.userc1'
//EXTENT ,nnnnnn
//LIBDEF CL,TO=userc1,FROM=cdmasc1,SEARCH=ciln
//DLBL   IDMSPCB,'work',,SD
//EXTENT SYS020,nnnnnn,,ssss,1111
//ASSGN  SYS020,disk
//EXEC   ESVSTNTC
Input SYSIDMS parameters, as required
Insert TNT input statements here
/*
*****
*
*          ASSEMBLE COMPILER OUTPUT
*
*****
//DLBL   IJSYSIN,'work',,SD
//EXTENT SYSIPT,nnnnnn,,ssss,1111
//ASSGN  SYSIPT,disk
//OPTION CATAL
//PHASE tntname,*
//EXEC ASMA90
//CLOSE SYSIPT,SYSRDR
/*
*****
*
*          LINK TNT TO CA-IDMS/DB CORE-IMAGE LIBRARY
*
*****
//EXEC LNKEDT
/&
```

4.4 Transaction name table

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, database, SYSIDMS, and other CA-IDMS files. ▶▶For a complete listing of IDMSLBLS, see Appendix H, “IDMSLBLS Procedure” on page H-1.
cdmscl	Filename of the CA-IDMS/DB core-image library
nnnnn	Volume serial number of disk unit
usercl	Filename of a user core-image library
user.usercl	File-id of a user core-image library
work	File-id of work file
ssss	Relative starting track of disk file
llll	Number of tracks required for disk file
disk	Device assignment of disk file
tntname	Name of the TNT PHASE
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

Chapter 5. Runtime Operations

5.1 Overview	5-3
5.2 Database preparation	5-4
5.2.1 Step 1: Install and prepare the DC/UCF system	5-4
5.2.2 Step 2: Prepare and install the CA-IDMS/DB database	5-4
5.2.3 Step 3: Create and compile the control tables	5-5
5.2.4 Step 4: Migrate data to the CA-IDMS/DB database	5-5
5.2.4.1 Sample migration utility JCL — OS/390	5-5
5.2.4.2 Sample migration utility JCL — VSE/ESA	5-8
5.3 Application preparation	5-11
5.3.1 Step 1: Modify the application JCL	5-11
5.3.1.1 Modify the application JCL — OS/390	5-11
5.3.1.2 Modify the application JCL — VSE/ESA	5-18
5.3.1.3 CA-IDMS/VSAM Transparency parameters	5-23
5.3.2 Step 2: Modify and recompile the application program	5-26
5.4 System execution	5-27
5.4.1 Step 1: Run the CA-IDMS/VSAM Transparency command interface	5-27
5.4.1.1 CA-IDMS/VSAM Transparency system commands	5-27
5.4.1.2 CA-IDMS/VSAM Transparency initialization — OS/390	5-28
5.4.1.3 CA-IDMS/VSAM Transparency initialization — VSE/ESA	5-28
5.4.2 Step 2: Bring up the DC/UCF system	5-29
5.5 Application execution	5-30
5.5.1 Normal termination	5-30
5.5.2 Abnormal termination	5-30
5.5.2.1 CA-IDMS/VSAM Transparency shutdown procedures — OS/390	5-31

5.1 Overview

Preparing for CA-IDMS/VSAM Transparency: CA-IDMS/VSAM Transparency accesses the CA-IDMS/DB database in the same way that CA-IDMS/DB application programs access the database. As a result, most of the preparation required to run CA-IDMS/VSAM Transparency is standard for the CA-IDMS/DB database environment.

►► For general information on database preparation and operation, refer to *CA-IDMS Database Administration*.

About this chapter: This chapter discusses the operational considerations that are specific to CA-IDMS/VSAM Transparency and are presented as follows:

- Database preparation
- Application preparation
- System execution
- Application execution

5.2 Database preparation

CA-IDMS/VSAM Transparency database preparation requires you to:

1. Install and prepare the CA-IDMS/DB central version or DC/UCF system (unless all CA-IDMS/VSAM Transparency applications are to be run in local mode).
2. Prepare the CA-IDMS/DB database.
3. Create and compile the control tables that CA-IDMS/VSAM Transparency will use at runtime.
4. Load the CA-IDMS/DB database.

The above steps are described in the discussions that follow.

5.2.1 Step 1: Install and prepare the DC/UCF system

Unless all CA-IDMS/VSAM Transparency application programs are run in local mode, you must prepare the central version or the DC/UCF system.

►► For information on system generation and startup, refer to *CA-IDMS System Generation* and *CA-IDMS System Operations*.

5.2.2 Step 2: Prepare and install the CA-IDMS/DB database

To prepare and install the CA-IDMS/DB database, you **must** define the following database elements:

- **Schema** — Define the records and sets that best represent the VSAM data set in the CA-IDMS/DB schema. Schema definitions are discussed in Chapter 2, “CA-IDMS/DB Schema and VSAM File Correspondences” on page 2-1.
- **DMCL modules** — Define and link edit DMCL modules into a load library so these modules are available at runtime. Information on loading DMCL modules can be found in *CA-IDMS Database Administration*.
- **Subschemas** — Define one or more subschemas, which can reside in either the dictionary load area or a load library. (Local mode requires the subschema to reside in a load library.)

For information on defining a subschema, and on transferring load modules to a load library, refer to *CA-IDMS Database Administration*.

Note: All areas that will be accessed by CA-IDMS/VSAM Transparency must be given a default usage mode. This usage mode must be defined in the subschema. If you are going to update the database through CA-IDMS/VSAM Transparency, be sure the default usage mode is an update usage mode.

5.2.3 Step 3: Create and compile the control tables

CA-IDMS/VSAM Transparency requires user-defined control tables that translate VSAM application program requests into CA-IDMS/DB database requests.

The FMT and TNT must be prepared, compiled, and linked as part of the preparation to run CA-IDMS/VSAM Transparency.

►► Instructions for preparing and compiling the control tables can be found in Chapter 4, “Preparing Control Information” on page 4-1.

5.2.4 Step 4: Migrate data to the CA-IDMS/DB database

Convert data structures using the migration utility: You can use the CA-IDMS/VSAM Transparency migration utility (ESVSMIGR) to migrate VSAM data to a CA-IDMS/DB database. This utility works with CA-IDMS/VSAM Transparency to convert KSDS, ESDS, RRDS, and alternate index data structures to CA-IDMS/DB data.

Steps to follow: To use the CA-IDMS/VSAM Transparency migration utility:

1. Use the FORMAT utility statement to initialize the area where the converted data will reside. (The FORMAT utility statement is described in *CA-IDMS Utilities*) If the area has already been initialized, you do not have to follow this step.

Note: You can load VSAM data into *any* standard CA-IDMS/DB area, including an existing area that already contains data.

2. Run the migration utility, using the FMT defined for the file to be migrated. If you have not already defined an appropriate FMT, refer to Chapter 4, “Preparing Control Information” on page 4-1, for instructions.

5.2.4.1 Sample migration utility JCL — OS/390

The OS/390 JCL used to migrate VSAM data to the CA-IDMS/DB database under the central version and in local mode is shown below.

Central version ESVSMIGR (OS/390)

```
//STEPCONV EXEC PGM=ESVSMIGR,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//SYSCTL DD DSN=idms.sysctl,DISP=SHR
//dcmmsg DD DSN=idms.sysmsg.dd1dcmmsg,DISP=SHR
//INPUT DD DSN=vsam.file,DISP=SHR
//OUTPUT DD SUBSYS=(ESVS,'FMT=fmtname','SUBSCHEMA=ssname',
// 'RBUFSZ=nnnnn')
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
```

Insert SYSIDMS parameters, as appropriate

<code>idms.dba.loadlib</code>	Data set name of the load library containing DMCL and database name table load modules
<code>idms.loadlib</code>	Data set name of the load library containing the CA-IDMS executable modules
<code>idms.sysctl</code>	Data set name of the CA-IDMS/DB SYSCTL file
<code>dcmsg</code>	DDname of the system message (DDLDCMSG) area
<code>idms.sysmsg.ddldcmsg</code>	Data set name of the system message (DDLDCMSG) area
<code>vsam.file</code>	Data set name of the VSAM file to be migrated
<code>fmtname</code>	Name of the file management table for the file to be migrated
<code>ssname</code>	Subschema name
<code>nnnnn</code>	Size of the buffer that CA-IDMS/VSAM Transparency will use for communication between the front end and back end; if not specified, the default is 512 bytes
<code>SYSIDMS</code>	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

Local mode ESVSMIGR (OS/390)

```
//STEPCONV EXEC PGM=ESVSMIGR,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//dcdm1 DD DSN=idms.system.ddldm1,DISP=SHR
//dcld DD DSN=idms.system.ddldcld,DISP=SHR
//dclog DD DSN=idms.system.ddldclog,DISP=SHR
//dcmsg DD DSN=idms.system.ddldcmsg,DISP=SHR
//SYSJRNL DD DUMMY
//userdb DD DSN=user.userdb,DISP=SHR
```

Additional DD statements as required

```
//INPUT DD DSN=vsam.file,DISP=SHR
//OUTPUT DD SUBSYS=(ESVS,'FMT=fmtname','SUBSCHEMA=ssname','RBUFFSZ=nnnnn')
//
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
```

Insert SYSIDMS parameters, as appropriate

idms.dba.loadlib	Data set name of the CA-IDMS/DB load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing the CA-IDMS executable modules
dcdml	DDname of the system dictionary definition (DDLML) area
idms.system.ddlml	Data set name of the system dictionary definition (DDLML) area
dclod	DDname of the system dictionary definition load (DDLDCLOD) area
idms.system.ddlclod	Data set name of the system dictionary definition load (DDLDCLOD) area
dclog	DDname of the system log (DDLDCLOG) area
idms.system.ddlclod	Data set name of the system log (DDLDCLOG) area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddlclod	Data set name of the system message (DDLDCMSG) area
userdb	DDname of the user database file to contain the migrated data
user.userdb	Data set name of the user database file to contain the migrated data
vsam.file	Data set name of the VSAM file to be migrated
fmtname	Name of the file management table for the file to be migrated
ssname	Subschema name
nnnnn	Size of the buffer that CA-IDMS/VSAM Transparency will use for communication between the front end and back end; if not specified, the default is 512 bytes. The value must accommodate the sum of the longest record length plus its key.
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

5.2.4.2 Sample migration utility JCL — VSE/ESA

The VSE/ESA JCL used to migrate VSAM data to the CA-IDMS/DB database under the central version and in local mode is shown below.

Central version ESVSMIGR (VSE/ESA)

```
// EXEC PROC=IDMSLBL5
// DLBL   usercl,'user.usercl'
// EXTENT ,xxxxxx
// LIBDEF CL,SEARCH=(cdmscl,usercl)
// DLBL   INPUT,'vsam.file',,VSAM
// EXTENT ,xxxxxx
// DLBL   OUTPUT,'FMT=fmtname',,VSAM,,CAT=ESVS
// DLBL   ESVS,'SUBSCHEMA=ssname,RBUFSZ=nnnnn',,VSAM
// ASSGN  SYSLST,PRINTER
// EXEC   ESVSMIGR
```

Insert SYSIDMS parameters, as required

/&

IDMSLBL5	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, SYSIDMS, parameter file and other files. ▶▶For a complete listing of IDMSLBL5, see Appendix H, “IDMSLBL5 Procedure” on page H-1.
xxxxxx	Volume serial number of disk unit
usercl	Filename of the user core-image library that contains the ESVSMIGR program
user.usercl	File-id of the user core-image library that contains the ESVSMIGR program
vsam.file	File-id of the VSAM file to be migrated
fmtname	Name of the file management table for the file to be migrated
ssname	Subschema name
nnnnnn	Size of the buffer that CA-IDMS/VSAM Transparency will use for communication between the front end and back end; if not specified, the default is 512 bytes
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

Local mode ESVSMIGR (VSE/ESA)

```
// EXEC PROC=IDMSLBS
// ASSGN SYSnnn,IGN
// DLBL cdmscl,'cdms.cdmscl'
// EXTENT ,xxxxxx
// DLBL usercl,'user.usercl'
// EXTENT ,xxxxxx
// LIBDEF CL,SEARCH=(cdmscl,usercl)
// DLBL userdb,'user.userdb',,SD
// EXTENT ,xxxxxx
```

Additional DLBL statements as required

```
// DLBL INPUT,'vsam.file',,VSAM
// EXTENT ,xxxxxx
// DLBL OUTPUT,'FMT=fmtname',,VSAM,,CAT=ESVS
// DLBL ESVS,'SUBSCHEMA=ssname,RBUFSZ=nnnnn',,VSAM
// ASSGN SYSLST,PRINTER
// EXEC ESVSMIGR
```

Insert SYSIDMS parameters, as required

```
/&
```

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, SYSIDMS, parameter file and other files. ▶▶For a complete listing of IDMSLBLS, see Appendix H, “IDMSLBLS Procedure” on page H-1.
nnn	Logical unit assignment for the SYSJRNL in the DMCL
cdmscl	Filename of the CA-IDMS/DB core-image library
cdms.cdmscl	File-id of the CA-IDMS/DB core-image library
xxxxxx	Volume serial number of disk unit
usercl	Filename of the user core-image library that contains the ESVSMIGR program
user.usercl	File-id of the user core-image library that contains the ESVSMIGR program
userdb	Filename of the user database file to contain the migrated data
user.userdb	File-id of the user database file to contain the migrated data
vsam.file	File-id of the VSAM file to be migrated
fmtname	Name of the file management table for the file to be migrated
ssname	Subschema name
nnnnnn	Size of the buffer that CA-IDMS/VSAM Transparency will use for communication between the front end and back end; if not specified, the default is 512 bytes
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

5.3 Application preparation

CA-IDMS/VSAM Transparency application preparation requires you to:

1. Modify the JCL of each job that will be processed by CA-IDMS/VSAM Transparency.
2. Modify and recompile VSAM application programs if any features used are not supported by CA-IDMS/VSAM Transparency.

5.3.1 Step 1: Modify the application JCL

Two DD or DLBL statements for each file: Each ddname (OS/390) or filename (VSE/ESA) must be unique for an application program. This means that in local mode, you will have two DD or DLBL statements for each file:

- One statement must describe the ddname or filename specified in the assembler ACB, COBOL SELECT statement, or PL/I DECLARE statement.
- The other statement must describe the external file name specified in the CA-IDMS/DB DMCL.

Example: Suppose a VSAM application program contains this COBOL SELECT statement:

```
SELECT CUSTFILE ASSIGN TO SYS030.
```

And suppose the DMCL definition of the VSAM data structure contains this file assignment:

```
ADD FILE CUSTOMER-FILE ASSIGN TO CSTFILE.
```

The JCL used to run the VSAM application in local mode will include one DD or DLBL statement for each of the files described above:

```
//SYS030 DD SUBSYS=(ESVS,'FMT=fmtname','SUBSCHEMA=ssname', ... )
//CSTFILE DD DSN=CUST-FILE,DISP=SHR
```

Instructions for modifying the application JCL in OS/390 and VSE/ESA are presented below.

5.3.1.1 Modify the application JCL — OS/390

SUBSYS parameter: In OS/390, CA-IDMS/VSAM Transparency is a subsystem and each CA-IDMS/VSAM Transparency file is considered a subsystem data set. This means that each data definition card for an CA-IDMS/VSAM Transparency file must contain the SUBSYS parameter.

Central version The JCL used to run a CICS CA-IDMS/VSAM Transparency application program under the central version is shown below. **CICS (OS/390)**

5.3 Application preparation

```
//ESVSCICS EXEC PGM=DFHSIP
//STEPLIB DD DSN=cics.system.loadlib1,DISP=SHR
//DFHRPL DD DSN=cics.system.loadlib,DISP=SHR
// DD DSN=cics.system.loadlib2,DISP=SHR
// DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
// DD DSN=user.loadlib, DISP=SHR
//dcmsg DD DSN=idms.sysmsg.dd1dcmsg,DISP=SHR
//sysctl DD DSN=user.sysctl,DISP=SHR
//userdd DD SUBSYS=(ESVS,'FMT=fmtname','SUBSCHEMA=ssname'
// 'MODE=CICS','RBUFSZ=nnnnn','TNT=tntname'
// 'CWADISP=disp')
// DSN=dataset-name,DISP=SHR
//SYSIDMS DD *
```

Insert SYSIDMS parameters, as appropriate

<code>user.loadlib</code>	Data set name of the user load library
<code>userdd</code>	DDname of the CA-IDMS/VSAM Transparency file, as it appears in the application
<code>cics.system.loadlib, loadlib1, loadlib2</code>	Names of the CICS system load libraries
<code>idms.sysmsg.ddldcmsg</code>	Data set name of the system message (DDLDCMSG) area
<code>idms.dba.loadlib</code>	Data set name of the load library containing the DMCL and database name table load modules
<code>idms.loadlib</code>	Data set name of the load library containing the CA-IDMS executable system modules
<code>dcmsg</code>	DDname of the system message (DDLDCMSG) area
<code>fmtname</code>	Name of the file management table that defines the escaped file
<code>ssname</code>	Subschema name
<code>nnnnnn</code>	Buffer size
<code>tnname</code>	Name of the transaction name table; optional when only one subschema exists for use with CA-IDMS/VSAM Transparency programs; required if more than one subschema exists
<code>disp</code>	Value equal to the CWADISP specified in the assembly of IDMSINTC
<code>data set name</code>	Name of a VSAM KSDS file defined in the VSAM catalog. This dummy file should be a KSDS dataset, even if the file being escaped was an ESDS file, to prevent VSAM catalog management from doing special processing associated with ESDS datasets. It should contain one or more records to prevent VSAM from opening the file I/O. CA-IDMS/VSAM Transparency does not require any special naming convention. The DSNAMES parameter is not needed with CICS Releases prior to Release 1.7.
<code>SYSIDMS</code>	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

►► For more information on these and additional CA-IDMS/VSAM Transparency parameters, refer to 5.3.1.3, “CA-IDMS/VSAM Transparency parameters” on page 5-23 located later in this section.

Example: The following statements illustrate the JCL used to run a sample CICS system that uses two CA-IDMS/VSAM Transparency files:

```
//ESVSCICS EXEC PGM=DFHSIP
//STEPLIB DD DSN=CICS.SYSTEM.LOADLIB1, DISP=SHR
//DFHRPL DD DSN=CICS.SYSTEM.LOADLIB, DISP=SHR
// DD DSN=CICS.SYSTEM.LOADLIB2, DISP=SHR
// DD DSN=IDMS.DBA.LOADLIB, DISP=SHR
// DD DSN=IDMS.LOADLIB, DISP=SHR
// DD DSN=USER.LOADLIB, DISP=SHR
//dcmmsg DD DSN=IDMS.SYSMSG.DDLDCMSG, DISP=SHR
//sysctl DD DSN=USER.SYSCTL, DISP=SHR
//SYSIDMS DD *
//EMPFILE DD SUBSYS=(ESVS, 'FMT=EMPFMT', 'SUBSCHEMA=EMPSS01',
// 'MODE=CICS', 'RBUFSZ=500', 'TNT=CICSTNT', 'CWADISP=16'),
// DSN=VSAM.ESVS.EMPFILE, DISP=SHR
//DEPFILE DD SUBSYS=(ESVS, 'FMT=DEPFMT')
// DSN=VSAM.ESVS.DEPFILE, DISP=SHR
```

The JCL used to run a batch CA-IDMS/VSAM Transparency application program under the central version and in local mode are shown below.

Central version (OS/390)

```
//userpgm EXEC PGM=userpgm
//STEPLIB DD DSN=user.loadlib, DISP=SHR
// DD DSN=idms.dba.loadlib, DISP=SHR
// DD DSN=idms.loadlib, DISP=SHR
//dcmmsg DD DSN=idms.sysmsg.ddldcmmsg, DISP=SHR
//sysctl DD DSN=user.sysctl, DISP=SHR
//userdd DD SUBSYS=(ESVS, 'FMT=fmtname', 'SUBSCHEMA=ssname'
// 'MODE=modetype', 'RBUFSZ=nnnnn')
//SYSIDMS DD *
```

Insert SYSIDMS parameters, as appropriate

userpgm	Application program name
user.loadlib	Data set name of the user load library
cics.system.loadlib, loadlib1, loadlib2	Names of the CICS system load libraries
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing the CA-IDMS executable system modules
dcmsg	DDname of the system message (DDLDCMSG) area
sysctl	DDname of the user SYSCTL file
user.sysctl	Data set name of the user SYSCTL file
userdd	DDname of the CA-IDMS/VSAM Transparency file, as it appears in the application
fmtname	Name of the file management table that defines the escaped file
ssname	Subschema name
modetype	Optional indicator of BATCH or CICS mode; if not specified, the default is BATCH
nnnnn	Buffer size
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

▶▶ For more information on these and additional CA-IDMS/VSAM Transparency parameters, refer to 5.3.1.3, “CA-IDMS/VSAM Transparency parameters” on page 5-23 located later in this section.

Example: The following statements illustrate the JCL used to run a sample batch CA-IDMS/VSAM Transparency application program under the central version; this program uses one file and one subschema.

```
//EMPLIST EXEC PGM=EMPLIST
//STEPLIB DD DSN=user.loadlib,DISP=SHR
// DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//SYSCTL DD DSN=IDMS.SYSCTL,DISP=SHR
//DEPFILE DD SUBSYS=(ESVS,'FMT=EMPFMT','SUBSCHEMA=EMPSS01',
// 'RBUFSZ=nnnnn')
//dcmsg DDDSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSIDMS DD *
```

Local mode (OS/390)

```
//userpgm EXEC PGM=userpgm
//STEPLIB DD DSN=user.loadlib,DISP=SHR
// DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//dcdm1 DD DSN=idms.system.ddldcm1,DISP=SHR
//dc1od DD DSN=idms.system.ddldc1od,DISP=SHR
//dc1og DD DSN=idms.system.ddldc1og,DISP=SHR
//SYSJRNL DD DSN=idms.journal.local, DISP=(NEW, KEEP), unit=devtype
//userdb DD DSN=user.userdb.file,DISP=SHR
//userdd DD SUBSYS=(ESVS,'FMT=fmtname','SUBSCHEMA=ssname')
Additional database file specifications
//SYSIDMS DD *
```

Insert SYSIDMS parameters, as appropriate

userpgm	Application program name
user.loadlib	Data set name of the user load library
idms.dba.loadlib	Data set name of the load library containing the CA-IDMS DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing the CA-IDMS system executable modules
dcdml	DDname of the system dictionary definition (DDL DML) area
idms.system.ddldml	Data set name of the system dictionary definition (DDL DML) area
dclod	DDname of the system dictionary definition load (DDL DCLOD) area
idms.system.ddldclod	Data set name of the system dictionary definition load (DDL DCLOD) area
dclog	DDname of the system log (DDL DCLOG) area
idms.system.ddldclog	Data set name of the system log (DDL DCLOG) area
userdb	DDname of the user CA-IDMS/DB file
user.userdb	Data set name of the user CA-IDMS/DB file
userdd	DDname of the CA-IDMS/VSAM Transparency file, as it appears in the application
fmtname	Name of the file management table that defines the escaped file
ssname	Subschema name
nnnnnn	Buffer size
devtype	Disk or tape
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ▶▶For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

▶▶ For more information on these and additional CA-IDMS/VSAM Transparency parameters, refer to 5.3.1.3, “CA-IDMS/VSAM Transparency parameters” on page 5-23 located later in this section.

Example: The following statements illustrate the JCL used to run a sample batch CA-IDMS/VSAM Transparency application program in local mode:

```
//EMPLIST EXEC PGM=EMPLIST
//STEPLIB DD DSN=user.loadlib,DISP=SHR
// DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
// DD DSN=IDMS.EMPLOAD,DISP=SHR
//SYSJRNL DD DSN=idms.journal.local, DISP=(NEW, KEEP), unit=devtype
//MYDB DD DSN=MY.EMPDB.EMPFILE,DISP=SHR
//EMPFILE DD SUBSYS=(ESVS,'FMT=EMPFMT','SUBSCHEMA=EMPSS01')
```

additional DD statements to run VSAM application

5.3.1.2 Modify the application JCL — VSE/ESA

Setting up DLBL statements: The use of DLBL statements with CA-IDMS/VSAM Transparency has specific rules that do not correspond to standard VSE/ESA DLBL statement rules:

First statement

■ **To set up the first DLBL statement:**

1. Specify 'FMT=fmtname' in place of the file-id. Fmtname is the name of the file management table that defines the escaped file.
2. Specify CAT=ESVS at the end of the DLBL statement.

A sample DLBL statement is shown below:

```
// DLBL EMPESC,'FMT=EMPFMT',,VSAM,,CAT=ESVS
```

Second statement

■ **To set up the second DLBL statement:**

1. Specify ESVS in place of the filename.
2. Specify a string of CA-IDMS/VSAM Transparency parameters in place of the file-id. The parameters must be separated by commas; the string must be enclosed in quotes and can contain no more than 44 characters (including the commas, but not including the quotes).
3. Specify CAT=ESVS1 if you want to continue the string of CA-IDMS/VSAM Transparency parameters on the third DLBL statement.

A sample DLBL statement is shown below:

```
// DLBL ESVS,'SUBSCHEMA=EMPSS01,MODE=BATCH',,VSAM,,CAT=ESVS1
```

All subsequent statements

■ **To set up all subsequent DLBL statements:**

1. Specify `ESVSn` in place of the filename, where `n` is a number from 1 to 9. `ESVSn` must be the same as the value specified in the `CAT` parameter of the previous `DLBL` statement.
2. Specify `CAT=ESVSn+1` if you want to continue the string of CA-IDMS/VSAM Transparency parameters on the next `DLBL` statement.

A sample `DLBL` statement is shown below:

```
// DLBL  ESVS1,'RBUFSZ=500',,VSAM,,CAT=ESVS2
```

Example: The `DLBL` statements required to define one CA-IDMS/VSAM Transparency file and its associated parameters might be set up as follows:

```
// DLBL  EMPESC,'FMT=EMPFMT',,VSAM,,CAT=ESVS
// DLBL  ESVS,'SUBSCHEMA=EMPSS01,MODE=BATCH',,VSAM,,CAT=ESVS1
// DLBL  ESVS1,'RBUFSZ=500',,VSAM,,CAT=ESVS2
// DLBL  ESVS2,'TRACE=0500,WRAP'
```

CICS: The JCL required to run a CICS system running CA-IDMS/VSAM Transparency application programs is shown below. **CICS (VSE/ESA)**

```
// DLBL userdd,'IDMS.CICS.VSAMT',,VSAM
// DLBL SYSESVS,'sysesvs.parms'
// EXEC DFHSIP
/&
```

<code>userdd</code>	Filename of the CA-IDMS/VSAM Transparency file, as it appears in the application
<code>IDMS.CICS.VSAMT</code>	Name of a VSAM dataset that is defined in the VSAM catalog. This dummy file should be a KSDS dataset, even if the file being escaped was an ESDS file, to prevent VSAM catalog management from doing special processing associated with ESDS datasets. It should contain one or more records to prevent VSAM from opening the file I/O.
<code>sysesvs.parms</code>	File ID of the SYSEVS parameters file

►► See 5.3.1.3, “CA-IDMS/VSAM Transparency parameters” on page 5-23, later in this section, for more information about CA-IDMS/VSAM Transparency parameters. See Appendix G, “VSE/ESA CICS SYSESVS Parameters” on page G-1 for more information about CICS SYSESVS parameters.

Example: The following statements illustrate the JCL used to run a sample CICS system running CA-IDMS/VSAM Transparency applications; this program uses two CA-IDMS/VSAM Transparency files:

```
// DLBL EMPFILE,'IDMS.CICS.VSAMT',,VSAM
// DLBL DEPFILE,'IDMS.CICS.VSAMT',,VSAM
// DLBL SYSESVS,'sysesvs.parms'
// EXEC DFHSIP
/&
```

The JCL required to run a batch CA-IDMS/VSAM Transparency application program under the central version and in local mode are shown below.

Central version (VSE/ESA)

```
// EXEC PROC=IDMSLBLS
// DLBL   usercl,user.usercl'
// EXTENT ,xxxxxx
// LIBDEF CL,SEARCH=(cdmscl,usercl)
// DLBL   userdd,'FMT=fmtname',,VSAM,,CAT=ESVS
// DLBL   ESVS,'SUBSCHEMA=ssname,MODE=modetype',,VSAM,,CAT=ESVS
// DLBL   ESVS1,'RBUFSZ=nnnn'
// EXEC   userpgm
```

Insert SYSIDMS parameters, as required

/&

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, SYSIDMS, parameter file and other files. ►►For a complete listing of IDMSLBLS, see Appendix H, “IDMSLBLS Procedure” on page H-1.
xxxxxx	Volume serial number of disk unit
usercl	Filename of a user core-image library
user.usercl	File-id of a user core-image library
userdd	Filename of the CA-IDMS/VSAM Transparency file, as it appears in the application
\$ESVS.FMT.FMTNAME	Name of a VSAM dataset that is defined in the VSAM catalog. You must specify ESVS.FMT. and then replace <i>fmtname</i> with your FMT name.
fmtname	Name of the file management table that defines the escaped file
ssname	Subschema name
modetype	Optional indicator of BATCH or CICS mode; if not specified, the default is BATCH
nnnnn	Buffer size
userpgm	Application program name
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

►► For more information on these and additional CA-IDMS/VSAM Transparency parameters, refer to 5.3.1.3, “CA-IDMS/VSAM Transparency parameters” on page 5-23 located later in this section.

Example: The following statements illustrate the JCL used to run a sample batch CA-IDMS/VSAM Transparency application program under the central version; this program uses one file:

```
// EXEC PROC=IDMSLBLS
// DLBL  USERCL,USER.USERCL'
// EXTENT ,SYSWK4
// LIBDEF CL,SEARCH=(CDMSCL,USERCL)
// DLBL EMPFILE,'FMT=EMPFMT',,VSAM,,CAT=ESVS
// DLBL ESVS,'SUBSCHEMA=EMPSS01,RBUFSZ=500'
// EXEC EMPLIST
```

```
dmc1=idmsdmc1
```

```
/&
```

Local mode (VSE/ESA)

```
// EXEC PROC=IDMSLBLS
// DLBL  usercl,user.usercl'
// EXTENT ,xxxxxx
// LIBDEF CL,SEARCH=(cdmscl,usercl)
// DLBL userdd,'FMT=fmtname',,VSAM,,CAT=ESVS
// DLBL ESVS,'SUBSCHEMA=ssname,MODE=modetype',,VSAM,,CAT=ESVS1
// DLBL ESVS1,'RBUFSZ=nnnn'
```

Additional database file specifications

```
// EXEC userpgm
```

Insert SYSIDMS parameters, as required

```
/&
```

IDMSLBLS	Name of the procedure provided at installation containing the file definitions for CA-IDMS dictionaries, databases, SYSIDMS, parameter file and other files. ►►For a complete listing of IDMSLBLS, see Appendix H, “IDMSLBLS Procedure” on page H-1.
nnn	Logical unit assignment for the SYSJRNL in the DMCL
cdmscl	Filename of the CA-IDMS/DB core-image library
cdms.cdmscl	File-id of the CA-IDMS/DB core-image library
xxxxxx	Volume serial number of disk unit
usercl	Filename of a user core-image library
user.usercl	File-id of a user core-image library
userdd	Filename of the CA-IDMS/VSAM Transparency file, as it appears in the application
fmtname	Name of the file management table that defines the escaped file
ssname	Subschema name
Modetype	Optional indicator of BATCH or CICS mode; if not specified, the default is BATCH
nnnnnn	Buffer size
userpgm	Application program name
SYSIDMS	DDname of the parameter file provided by CA-IDMS to specify runtime directives and operating system-dependent parameters. ►►For a complete description of the SYSIDMS parameter file, see <i>CA-IDMS Database Administration</i> .

►► For more information on these and additional CA-IDMS/VSAM Transparency parameters, refer to 5.3.1.3, “CA-IDMS/VSAM Transparency parameters” on page 5-23 located later in this section.

Sample: The following statements illustrate the JCL used to run a sample batch CA-IDMS/VSAM Transparency application program in local mode:

```
// EXEC PROC=IDMSLBL5
// DLBL USERCL,'USER.USERCL'
// EXTENT ,SYSWK4
// LIBDEF CL,SEARCH=(CDMSCL,USERCL)
// DLBL EMPFILE,'FMT=EMPFMT',,VSAM,,CAT=ESVS
// DLBL ESVS,'SUBSCHEMA=EMPSS01,MODE=CICS',,VSAM,,CAT=ESVS1
// DLBL ESVS1,'RBUFSZ=500'
```

Additional DLBL statements required to run VSAM application

```
// EXEC EMLIST
dbname=empdb
/ &
```

5.3.1.3 CA-IDMS/VSAM Transparency parameters

The table below presents a detailed list of the CA-IDMS/VSAM Transparency parameters, which you can use in the application program JCL.

Considerations: The following considerations apply:

- Unless otherwise specified, the parameters in this table are optional.
- The FMT parameter must be included in all CA-IDMS/VSAM Transparency DD or DLBL statements for both OS/390 and VSE/ESA.
- All other parameters must be coded as follows:
 - In OS/390, the parameters can be included in any DD statement, but must appear only once per application.
 - In VSE/ESA, the parameters must be included in the DLBLfile statement that starts with ESVS.

Parameter	Description
CWADISP= <i>nnnn</i>	Specifies a value equal to the CWADISP specified in the assembly of IDMSINTC. The CWADISP parameter is mandatory for CICS processing and is invalid for batch processing. It can be up to 4 digits long and is typically around 16.
DBNAME= <i>database-name</i>	Specifies the default database to be used with the CA-IDMS/VSAM Transparency application program (BATCH mode) or the default database to be used with all CICS CA-IDMS/VSAM Transparency transactions (CICS mode). This parameter is used in a multiple-database environment. You can override this parameter for each CICS CA-IDMS/VSAM Transparency transaction by using the transaction name table. The DBNAME parameter can be from 1 to 8 characters long.

Parameter	Description
FMT= <i>fnt-name</i>	Identifies the file management table that defines the file to be escaped. The FMT parameter is mandatory for all CA-IDMS/VSAM Transparency files. It can be from 1 to 8 characters long and must be included in all file statements.
MSG=LOG/CONS/OFF	<p>Displays the CA-IDMS/VSAM Transparency key feedback (runtime) messages during program execution. The options are:</p> <ul style="list-style-type: none"> ■ LOG (the default) writes the key feedback messages to the log file. ■ CONS writes the key feedback messages to the operator console. ■ OFF turns off the display of key feedback messages. <p>CA-IDMS/VSAM Transparency writes key feedback messages for any open call, close call, or call that is in error. The format is as follows:</p> <p>FDBK=<u>nnnnnnnn</u></p> <p><i>Nnnnnnnn</i> is an 8-byte field that contains the key feedback message code.</p>
MODE= BATCH/CICS	Indicates whether the CA-IDMS/VSAM Transparency application program is to run in BATCH or CICS mode. The default is BATCH.
NODENAME= <i>nodename</i>	<p>Specifies the nodename to be used with the CA-IDMS/VSAM Transparency application program (BATCH mode) or the default nodename to be used with all CICS CA-IDMS/VSAM Transparency transactions (CICS mode). This parameter is used in a multiple-database environment. You can override this parameter for each CICS CA-IDMS/VSAM Transparency transaction by using the transaction name table.</p> <p>The NODENAME parameter can be from 1 to 8 characters long.</p>
OCMSGS=ALL/ERROR	<p>Specifies which messages you want displayed on the console. The options are:</p> <ul style="list-style-type: none"> ■ ALL indicates both informational and error messages are displayed on the console. ■ ERROR indicates that only error messages are displayed on the console.

Parameter	Description
RBUFSZ=nnnnn	<p>Indicates the size of the buffer that CA-IDMS/VSAM Transparency will use for communication between the front end and the back end. The default is 512 bytes.</p> <p>The RBUFSZ parameter can be from 1 to 5 digits long and is valid for the central version only.</p> <p>The value must accommodate the sum of the longest record length plus its key length. CA-IDMS/VSAM Transparency adds the length of its overhead to the RBUFSZ you specify.</p>
STATS= ON/FILE	<p>Requests that database statistics be collected for all CA-IDMS/VSAM Transparency files accessed either by the CA-IDMS/VSAM Transparency application program (BATCH mode) or by all CA-IDMS/VSAM Transparency CICS transactions (CICS mode). The options are:</p> <ul style="list-style-type: none"> ■ ON places the statistics for all CA-IDMS/VSAM Transparency files together on the system log. ■ FILE places the statistics for each CA-IDMS/VSAM Transparency file separately on the system log. <p>The STATS parameter is not available in local mode.</p>
SUBSCHEMA= <i>subschema-name</i>	<p>Identifies the subschema to be used with the CA-IDMS/VSAM Transparency application program (BATCH mode) or the default subschema to be used with all CA-IDMS/VSAM Transparency CICS transactions (CICS mode). This parameter is mandatory for both BATCH mode and CICS mode. You can override the SUBSCHEMA parameter for each CICS CA-IDMS/VSAM Transparency transaction by using the transaction name table.</p> <p>The SUBSCHEMA parameter can be from 1 to 8 characters long.</p>

Parameter	Description
TNT= <i>tnt-name</i>	<p>Identifies the transaction name table that CA-IDMS/VSAM Transparency will use for processing CICS transactions. This parameter is optional if all transactions are to use the same subschema. It is required if more than one subschema is to be used. The TNT parameter is invalid in the batch environment.</p> <p>The TNT parameter can be from 1 to 8 characters long.</p>
TRACE= <i>nnnn</i> ,WRAP/LOG	<p>Requests that a trace of calls made by the CA-IDMS/VSAM Transparency application program be displayed on the system log. <i>Nnnn</i> is the number of fullwords allowed in the trace table and can be from 1 to 4 digits long. The TRACE options are as follows:</p> <ul style="list-style-type: none">■ WRAP (the default) enables the trace table to wrap around if the fullword limit is reached. This means that subsequent program calls will be recorded over the existing trace table.■ LOG writes the trace table to CDMSLOG and then clears the trace table, if the limit for the trace table is reached. This means that subsequent program calls will be recorded on the cleared trace table. The LOG option is not available in local mode.

5.3.2 Step 2: Modify and recompile the application program

If any features used in the application program are not supported by CA-IDMS/VSAM Transparency, you must modify and recompile the application program. To modify and recompile the application program, follow the instructions presented in the VSAM documentation.

5.4 System execution

Requirements: CA-IDMS/VSAM Transparency system execution requires you to:

1. Run the CA-IDMS/VSAM Transparency command interface with the proper commands to start up CA-IDMS/VSAM Transparency.
2. Bring up the central version (unless running in local mode).

You must perform these steps each time the operating system is cycled, before you run any CA-IDMS/VSAM Transparency jobs. You can perform these steps in any order, but you must perform them before bringing up CICS.

5.4.1 Step 1: Run the CA-IDMS/VSAM Transparency command interface

A list of CA-IDMS/VSAM Transparency system commands is presented below, followed by separate instructions for running the CA-IDMS/VSAM Transparency command interface in OS/390 and VSE/ESA.

5.4.1.1 CA-IDMS/VSAM Transparency system commands

Considerations: The following considerations apply to CA-IDMS/VSAM Transparency system commands:

- In OS/390, you enter these commands directly from an operator console.
- In VSE/ESA, you can enter these commands in one of two ways:
 - You can run ESVSINIT with these parameters. The END parameter must always be the last statement included in the job stream for ESVSINIT.
 - You can enter these commands from the operator console if you use the CONSOLE command.

System command descriptions: The CA-IDMS/VSAM Transparency system commands are described below:

- **CONSOLE** (VSE/ESA only) redirects the relay of operator commands from SYSIPT to the operator console.
- **DISPLAY** causes all active CA-IDMS/VSAM Transparency jobs to be displayed on the operator console.
- **END** causes the CA-IDMS/VSAM Transparency command interface to terminate *without affecting the status of CA-IDMS/VSAM Transparency*.
- **SHUTDOWN** prevents new users from accessing CA-IDMS/VSAM Transparency and brings down CA-IDMS/VSAM Transparency when the last current user is finished.

- **SHUTDOWN,I** causes CA-IDMS/VSAM Transparency to terminate immediately. As a result, any jobs that are currently using CA-IDMS/VSAM Transparency are aborted and their updates backed out.
 - ▶▶ See 5.5, “Application execution” on page 5-30, later in this manual, for more information about abnormal termination.
- **START** brings up CA-IDMS/VSAM Transparency and causes it to be initialized. START is valid only when CA-IDMS/VSAM Transparency is INACTIVE.
- **STATUS** shows the current status of CA-IDMS/VSAM Transparency.

5.4.1.2 CA-IDMS/VSAM Transparency initialization — OS/390

Startup procedure: For OS/390, the CA-IDMS/VSAM Transparency command interface startup procedure must reside in SYS1.PROCLIB. For information on the command interface startup procedure, refer to Appendix B, “CA-IDMS/VSAM Transparency Installation” on page B-1.

To run the startup procedure in OS/390, perform these steps from an operator console:

Step 1

```
s esvs
```

[Enter]

```
nn ES227002: VSAM/T INACTIVE, ENTER REPLY
```

Step 2

```
R nn,start
```

[Enter]

```
nn ES227001: VSAM/T ACTIVE, ENTER REPLY
```

Step 3

```
R nn,end
```

[Enter]

▶▶ A complete list of CA-IDMS/VSAM Transparency command interface messages for the OS/390 environment is presented in Appendix D, “CA-IDMS/VSAM Transparency Return Codes and Messages” on page D-1.

5.4.1.3 CA-IDMS/VSAM Transparency initialization — VSE/ESA

Startup procedure: For VSE/ESA, the CA-IDMS/VSAM Transparency command interface startup program may be in any private core-image library. If a dynamic SVC is to be used, IDMSSESVC must be run prior to the startup program.

To initiate the CA-IDMS/VSAM Transparency system in VSE/ESA, execute the job stream shown below: **ESVSINIT (VSE/ESA)**

```
// DLBL idmslib,'idms.lib'
// EXTENT ,xxxxxx
// LIBDEF PHASE, SEARCH=idmslib.sublib
// ASSGN SYSLST, PRINTER
// ASSGN SYSRDR, READER
// EXEC ESVSINIT
START
END
/*
/ &
```

idmslib	Filename of the CA-IDMS Library
idms.lib	File-id of the CA-IDMS Library
xxxxxx	Volume serial number of disk unit

Note that parameters specified as input to ESVSINIT must start in column 1.

Return messages: For a startup that proceeds normally, the CA-IDMS/VSAM Transparency command interface will return the following messages:

```
ES227107: VSAM/T INITIALIZATION STARTED
ES227101: VSAM/T ACTIVE
          VSAM/T ACTIVE, ENTER REPLY
```

A complete list of CA-IDMS/VSAM Transparency command interface messages for the VSE/ESA environment is presented in Appendix D, “CA-IDMS/VSAM Transparency Return Codes and Messages” on page D-1.

5.4.2 Step 2: Bring up the DC/UCF system

You must bring up the DC/UCF system, unless all CA-IDMS/VSAM Transparency applications are to be run in local mode.

►► For instructions on bringing up a DC/UCF system, refer to *CA-IDMS System Operations*.

5.5 Application execution

At this point, you can execute your VSAM application by using the JCL described earlier in this section. Information on CA-IDMS/VSAM Transparency program termination and recovery responses is presented below.

5.5.1 Normal termination

Batch application: For a batch application, all CA-IDMS/VSAM Transparency files are accessed through a single run unit. When the application terminates normally or closes all CA-IDMS/VSAM Transparency files, CA-IDMS/VSAM Transparency finishes the run unit. Normal completion of a transaction results in updates being committed to the database.

Note: If the same job subsequently opens any CA-IDMS/VSAM Transparency file, a new run unit will begin.

CICS processing: Under CICS, each transaction accesses all CA-IDMS/VSAM Transparency files through its own run unit. When the transaction terminates normally or closes all CA-IDMS/VSAM Transparency files, CA-IDMS/VSAM Transparency finishes the run unit. Normal completion of a transaction results in updates being committed to the database.

5.5.2 Abnormal termination

VSAM error codes: Any error that occurs in CA-IDMS/VSAM Transparency is translated into a VSAM error code and is returned to the application.

Data is protected: Because CA-IDMS/VSAM Transparency is integrated with the CA-IDMS/DB database, data integrity is protected by CA-IDMS/DB's journaling and recovery services.

►► Information on backup and recovery procedures can be found in *CA-IDMS Database Administration*.

Considerations: The following considerations apply to the abnormal termination of an CA-IDMS/VSAM Transparency application program:

- For a CICS transaction, all CA-IDMS/VSAM Transparency updates are rolled back to a point before the CA-IDMS/VSAM Transparency files were processed.
- Under central version, all recovery is handled automatically by CA-IDMS/DB recovery procedures.
- In local mode, the database that contains the CA-IDMS/VSAM Transparency files must be restored manually.

5.5.2.1 CA-IDMS/VSAM Transparency shutdown procedures — OS/390

To shutdown CA-IDMS/VSAM Transparency, perform the following steps from an operator console:

Step 1

s esvs

[Enter]

nn ES227001: VSAM/T ACTIVE, ENTER REPLY

Step 2

R nn,shutdown

[Enter]

nn ES227002: VSAM/T INACTIVE, ENTER REPLY

Step 3

R nn,end

[Enter]

Appendix A. CA-IDMS/VSAM Transparency Architecture

A.1	About this appendix	A-3
A.2	CA-IDMS/VSAM Transparency architecture — batch processing	A-4
A.3	System services manager	A-5
A.4	Request processing modules	A-6
A.4.1	The CA-IDMS/VSAM Transparency front end	A-6
A.4.2	The CA-IDMS/VSAM Transparency back end	A-7
A.5	CA-IDMS/VSAM Transparency architecture — CICS processing	A-8
A.6	Control tables	A-10
A.6.1	File management table	A-10
A.6.2	Transaction name table	A-10

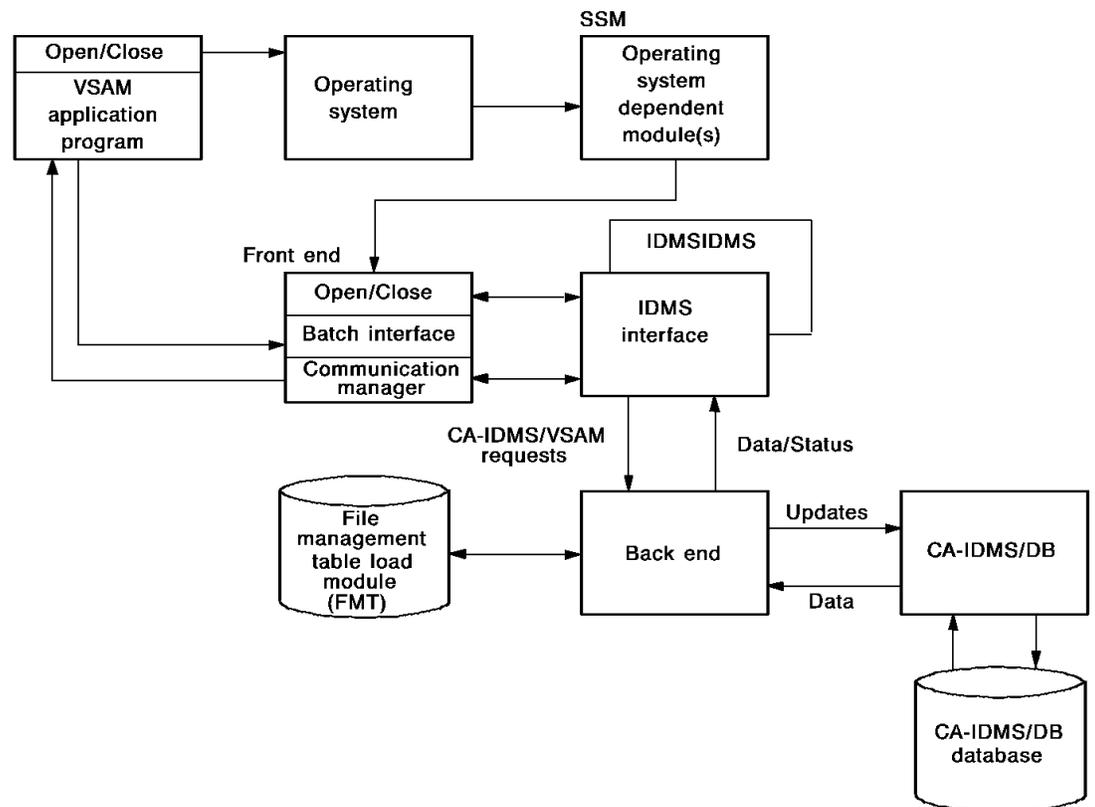
A.1 About this appendix

This appendix provides information about CA-IDMS/VSAM Transparency's components, including:

- The system services manager
- Front-end and back-end request processing modules
- The CA-IDMS/VSAM Transparency CICS interface
- Control tables

A.2 CA-IDMS/VSAM Transparency architecture — batch processing

The following diagram illustrates the CA-IDMS/VSAM Transparency components used for batch processing.



The functions of each CA-IDMS/VSAM Transparency module are presented in detail below.

A.3 System services manager

The CA-IDMS/VSAM Transparency system services manager provides operating-system-dependent services for CA-IDMS/VSAM Transparency files. When an application program issues a request to open or close a CA-IDMS/VSAM Transparency file, the system services manager performs the appropriate operating-system open or close function. The system services manager then transfers the request to the CA-IDMS/VSAM Transparency front-end module.

A.4 Request processing modules

CA-IDMS/VSAM Transparency processes VSAM requests by using two modules: the front end and the back end. These modules simulate VSAM functions and are operating system independent.

A.4.1 The CA-IDMS/VSAM Transparency front end

The CA-IDMS/VSAM Transparency front end is the application request processing module that does the following:

- **Initializes and terminates files** through the open/close processing routine. This routine:
 - Opens and closes files
 - Initializes and terminates the front-end control blocks at the job level (for batch processing), transaction level (for CICS processing), and file level (for either batch or CICS processing)
 - Calls the back-end module for back-end control block initialization or termination
- **Interfaces with batch applications** through the batch interface routine. This routine:
 - Receives batch process calls (GET, PUT, POINT, ERASE, ENDREQ, CHECK)
 - Saves the user environment
 - Sets up the CA-IDMS/VSAM Transparency environment
 - Calls the communication manager routine

For CICS, the batch interface is functionally replaced by the CA-IDMS/VSAM Transparency CICS interface.

- **Handles all communication between the VSAM application and the back end** through the communication manager routine. This routine:
 - Validates application requests
 - Transmits processing requests to the back-end module
 - Receives data and status information from the back end and transmits this information back to the VSAM application

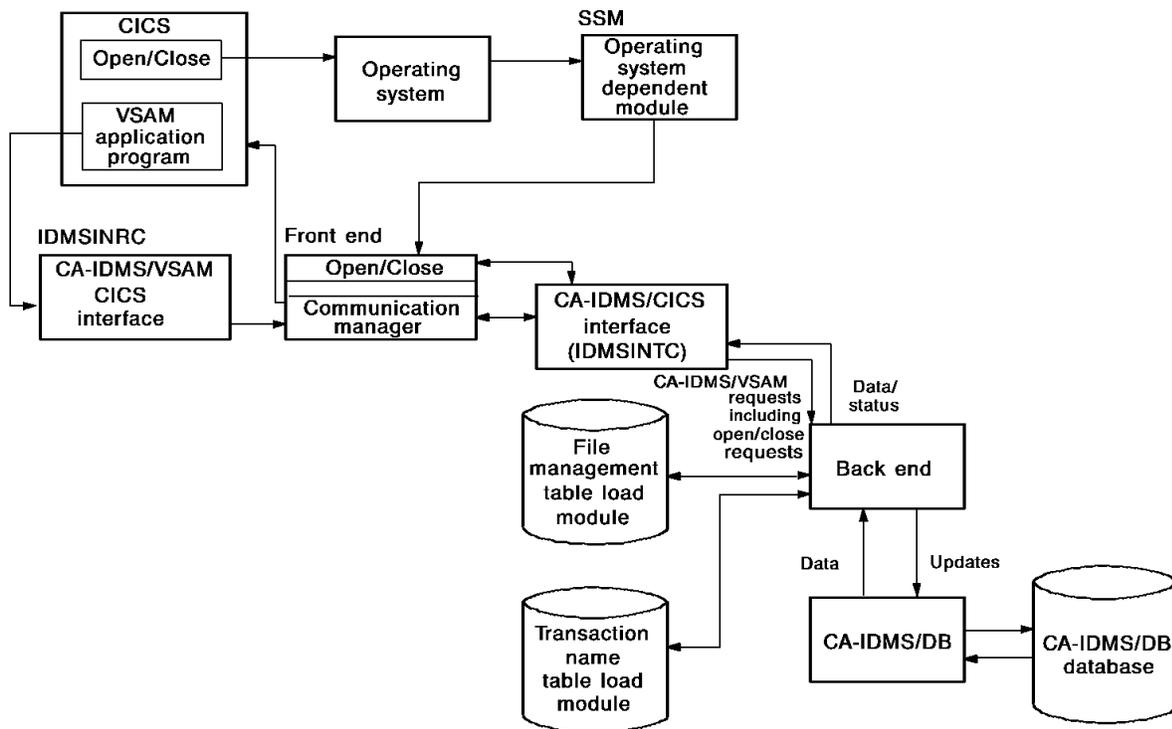
A.4.2 The CA-IDMS/VSAM Transparency back end

The CA-IDMS/VSAM Transparency back end accepts requests from the front end and translates VSAM requests to CA-IDMS/DB database calls. The back end does the following:

- **Initializes the run unit.** The back end:
 - Validates open and close requests.
 - Initializes and terminates the back-end control blocks. This includes loading the file management tables, which establish correspondences between VSAM requests and the database, and loading the transaction name table, which establishes correspondences between CICS transactions and CA-IDMS/DB subschemas.
- **Translates VSAM requests.** The back end:
 - Processes VSAM request parameter lists (RPLs)
 - Uses the file management table to convert VSAM requests to the corresponding DML statements
- **Accesses the database.** The back end:
 - Issues standard DML calls to access the database
 - Receives requested data from the database, including CA-IDMS/DB status codes
 - Converts database output to VSAM format
 - Sends the VSAM data and status information to the front-end communication manager routine, which then returns the output to the application program

A.5 CA-IDMS/VSAM Transparency architecture — CICS processing

The following diagram illustrates the CA-IDMS/VSAM Transparency components used for CICS processing.



CICS uses additional components: CICS CA-IDMS/VSAM Transparency processing uses all of the request processing routines described above, except for the batch interface routine. CICS CA-IDMS/VSAM Transparency processing also uses these additional components:

- The **CA-IDMS/VSAM Transparency CICS interface** communicates directly with the front-end communication manager routine. Each CICS transaction uses its own CA-IDMS/VSAM Transparency run unit for efficient multithreaded processing, task-level journaling, and recovery.
- The **CA-IDMS CICS interface (IDMSINTC)** establishes communication between the CA-IDMS/VSAM Transparency front end and back end. Requests are transmitted to the back end; data and status code information are returned to the front end. IDMSINTC must be assembled with CA-IDMS/VSAM Transparency parameters.
- The **transaction name table** maps the transaction names of CICS applications to CA-IDMS/DB subschema names. This table is discussed later in this appendix under A.6, “Control tables” on page A-10.

CICS interface points: At runtime, CA-IDMS/VSAM Transparency interfaces with CICS at the following points:

- **Open/Close** When IDMSINTC is started, the opens are processed and then CA-IDMS/VSAM Transparency writes error messages showing the feedback code.
- **Transaction start initialization** Transaction start is handled by the CA-IDMS/VSAM Transparency CICS interface. This interface:
 - Uniquely identifies the transaction before passing control to CA-IDMS/VSAM Transparency
 - Initializes the CA-IDMS/VSAM Transparency environment for the transaction at the first VSAM request
 - Enables each CICS transaction to run as a separate CA-IDMS/DB run unit
 - Takes the place of the front-end batch interface routine. The CA-IDMS/VSAM Transparency CICS interface:
 - Saves the user environment
 - Sets up the CA-IDMS/VSAM Transparency environment for the individual transactions
 - Calls the front-end communication manager routine
- **Transaction end** IDMSINTC contains code that checks for normal or abnormal termination of transactions.
 - ▶▶ For information on the IDMSINTC macro and its parameters, refer to CA-IDMS installation manual for your operating system

A.6 Control tables

FMT and TNT control tables: There are two types of control tables in CA-IDMS/VSAM Transparency:

- The **file management table (FMT)** defines the correspondences between VSAM data structures and CA-IDMS/DB records and sets.
- The **transaction name table (TNT)** maps the transaction names of CICS applications to CA-IDMS/DB subschema names.

Each control table must be compiled separately and stored in a load library before runtime.

Table requirements: The following requirements apply to these tables:

- Control tables are compiled from user-supplied statements.
- Control tables are linked into the CA-IDMS/DB load library.
- Control tables are specified by name in the JCL.

A.6.1 File management table

Mandatory control table: The file management table is a mandatory control table. One FMT is required for each VSAM data set used, regardless of the processing environment (batch or CICS).

Contents: The FMT includes the following information:

- The table name
- The VSAM data set type
- The CA-IDMS/DB record name that corresponds to the VSAM record
- The CA-IDMS/DB set name that will contain the CA-IDMS/DB records

The FMT can also include key length, key position, relative-record number, record length, buffer size, and erase options.

►► For detailed information on the FMT, refer to Chapter 4, “Preparing Control Information” on page 4-1.

A.6.2 Transaction name table

Use for different subschemas: The transaction name table maps the CICS application transaction names to CA-IDMS/DB subschema names. If all CICS transactions to be run under CA-IDMS/VSAM Transparency are to use the same subschema, it is not necessary to create a TNT. Instead, CA-IDMS/VSAM Transparency will use the

subschema name coded in the JCL. If the CICS transactions are not to use the same subschema, a TNT must be created.

Required statements: The following user statements are required for a TNT:

- The TNT name
- The transaction name
- The subschema name

►► For detailed information on the TNT, refer to Chapter 4, “Preparing Control Information” on page 4-1.

Appendix B. CA-IDMS/VSAM Transparency Installation

B.1 About this appendix	B-3
B.2 OS/390 Installation	B-4
B.2.1 Installing CA-IDMS/VSAM Transparency in the CICS enviroment . . .	B-5
B.3 VSE/ESA Installation	B-7
B.3.1 Installing CA-IDMS/VSAM Transparency in the CICS environment . .	B-8

B.1 About this appendix

This appendix provides detailed information on the installation of CA-IDMS/VSAM Transparency in conjunction with the following operating systems:

- OS/390
- VSE/ESA

CICS considerations are addressed under the corresponding operating system.

B.2 OS/390 Installation

Steps to follow: To install CA-IDMS/VSAM Transparency under OS/390, complete the following steps:

1. **Install the CA-IDMS/VSAM Transparency software into the CA-IDMS/DB library** by using integrated installation procedures. These procedures are described in *CA-IDMS Installation and Maintenance Guide - OS/390*.
2. **Copy ESVSINIT, ESVSSBLD, and ESVSSSSM to an APF-authorized library** that will be used for the CA-IDMS/VSAM Transparency command interface startup procedure. (If the library used in Step 1 is APF authorized, you do not have to recopy these modules.)
3. **Include an ADD PROGRAM statement for ESVSAMBE** in the system definition of all DC/UCF systems to be used with CA-IDMS/VSAM Transparency. Specify the ADD PROGRAM statement as follows:

```
ADD PROGRAM ESVSAMBE
LANGUAGE IS ASSEMBLER
REENTRANT
RESIDENT.
```

Note: As part of the normal installation process for CA-IDMS/VSAM Transparency, your system definition will be updated with the definition of all programs required for this product. If you need to add these definitions to another system, source module DLODEVSM in your installed library contains the definition of all the CA-IDMS/VSAM Transparency programs.

►► For more information on the ADD PROGRAM statement, refer to *CA-IDMS System Generation*.

4. **Make additions to the CA-IDMS system for variable length records**, as detailed in Appendix C, “Variable-Length Record Considerations” on page C-1.
5. **Set up a CA-IDMS/VSAM Transparency command interface procedure** to start up the CA-IDMS/VSAM Transparency system. The procedure must be a member of SYS1.PROCLIB. (OS/390) requires *all* subsystem startup procedures to be in SYS1.PROCLIB.)

The name of this procedure is assumed to be ESVS in this manual. If you want to use different versions, for example one to test and one for production, then the name of the version on the PROC statement must match the name on the SUBSYS= control statement.

Startup JCL

To set up a startup procedure, use the JCL exactly as shown below. **ESVSINIT (OS/390)**

```
//ESVS PROC
//STEPESVS EXEC PGM=ESVSINIT,TIME=1440
//STEPLIB DD DSN=idms.apf.loadlib,DISP=SHR,VOL=SER=nnnn,
//          UNIT=unit
//APFLIB DD DSN=idms.apf.loadlib,DISP=SHR,VOL=SER=nnnn,
//          UNIT=unit
```

idms.apf.loadlib	Data set name of the APF-authorized CA-IDMS/DB load library that includes ESVSINIT and ESVSSSSM
nnnn	Volume serial number for the APF-authorized CA-IDMS/DB load library
unit	Disk type

6. **Define CA-IDMS/VSAM Transparency as a subsystem in OS/390.** You do this by adding the subsystem name, ESVS, to member IEFSSN00 of SYS1.PARMLIB.
7. **You can define ESVS as a subsystem by executing the program ESVSSBLD with *parm*=(*<ESVS>*).** In this way, you don't need to wait for a system IPL before using VSAM/T. ESVSSBLD must be on an authorized library, where *<ESVS>* is the chosen subsystem name. The job does not require dd statements. If the subsystem name has not been defined in SYS1.PARMLIB subsystem name member, then this program must be executed whenever an IPL of OS/390 has been done.
8. **Cycle the OS/390 system** to implement the definition of the CA-IDMS/VSAM Transparency subsystem.

Note: It is not necessary to cycle the system if you reinstall CA-IDMS/VSAM Transparency.

B.2.1 Installing CA-IDMS/VSAM Transparency in the CICS environment

Initial Installation: When installing CA-IDMS/VSAM Transparency in the CICS environment, CICSOPTS will be assembled and link edited to create IDMSINTC. This is done automatically if CA-IDMS/VSAM Transparency is installed as part of an integrated installation. All parameters for CICSOPTS that are required for the VSAM Transparency will be automatically generated by the CAISAG installation utility when you indicate the product is to be installed, either as part of an integrated install or as a single product during ADDON processing.

Modifying CICSOPTS: If you need to re-assemble CICSOPTS to change the installation options, you must edit the CICSOPTS member in your SMP/E PPOPTION file and apply a usermod causing the module to be reassembled. A sample usermod, UMCOPTS, can be found in the SAMP JCL library installed with your product tape.

Important: Be sure the PLT entries are created to execute IDMSINTC at CICS startup and shutdown. Additionally, be sure that PPT entries for CA-IDMS modules have been added to your CICS system. These are used by IDMSINTC in setting up the CA-IDMS environment.

Files to be accessed through the VSAM/T must be defined through the FCT assemblies to be initially closed, enabled so that CICS does not attempt to open them until IDMSINTC has been started.

In order for VSAM/T to work when IDMSINTC is defined in the PLT to be executed at startup, CA-IDMS CV must be brought up. If CICS is brought up before CA-IDMS CV is brought up, IDMSINTC must be started through a transaction code. Before starting IDMSINTC, take measures to prevent applications from attempting to access the VSAM/T files so that you can avoid OPEN errors.

►►For more information on adding PPT entries, see the CA-IDMS installation guide for your operating environment. For information on the IDMSINTC macro and its parameters, refer to *CA-IDMS System Operations*.

B.3 VSE/ESA Installation

Steps to follow: To install CA-IDMS/VSAM Transparency under VSE/ESA, complete the following steps:

1. **Install the CA-IDMS/VSAM Transparency software into the CA-IDMS/DB library** by using the integrated installation procedures.

►► These procedures are described in *CA-IDMS Installation and Maintenance Guide - VSE/ESA*.

2. **Install the CA-IDMS SVC** as described in CA-IDMS installation manual for your operating system.

►► If using variable-length CA-IDMS/VSAM Transparency records, there are additional installation considerations. See Appendix C, “Variable-Length Record Considerations” on page C-1 for a list of these considerations.

3. If you intend to reassemble ESVSSPVT *without* using the integrated installation procedures, assemble and link edit ESVSSPVT as SVA eligible.

Sample JCL Link Edit ESVSSPVT (VSE/ESA)

```
// DLBL cdms1,'cdms.cdms1'
// EXTENT ,xxxxxx
// DLBL cdmsc1,'cdms.cdmsc1'
// EXTENT ,xxxxxx
// LIBDEF SL,FROM=cdms1
// LIBDEF CL,TO=cdmsc1
// OPTION CATAL
// PHASE ESVSSPVT,*,SVA
// EXEC ASMA90
//     ESVSSPVT SVC=nnn
//     END
/*
// EXEC LNKEDT,SIZE=500K
/ &
```

cdms1	Filename of the CA-IDMS/DB source statement library
cdms.cdms1	File-id of the CA-IDMS/DB source statement library
xxxxxx	Volume serial number of disk unit
cdmsc1	Filename of the CA-IDMS/DB or user core-image library
cdms.cdmsc1	File-id of the CA-IDMS/DB or user core-image library
nnn	This is the number of the CA-IDMS SVC

4. Add the following phases to the System Directory List:

- ESVSDOSM,SVA
- ESVSCBMM,SVA

- ESVSSPVT,SVA

You can add these phases in one of two ways:

- Issue a SET SDL command after the operating system is cycled and prior to executing CA-IDMS/VSAM Transparency startup.
- Add the phases to the automatic system initialization procedure (ASIPROC).

5. **Include an ADD PROGRAM statement for ESVSAMBE** in the system definition of all DC/UCF systems to be used with CA-IDMS/VSAM Transparency. Specify the ADD PROGRAM statement as follows:

```
ADD PROGRAM ESVSAMBE
  LANGUAGE IS ASSEMBLER
  REENTRANT
  RESIDENT.
```

Note: As part of the normal installation process for CA-IDMS/VSAM Transparency, your system definition will be updated with the definition of all programs required for this product. If you need to add these definitions to another system, source module DLODEVSM in your installed library contains the definition of all the CA-IDMS/VSAM Transparency programs.

►► For more information on the ADD PROGRAM statement, refer to *CA-IDMS System Generation*.

6. **Make additions to the CA-IDMS system for variable length records**, as detailed in Appendix C, “Variable-Length Record Considerations” on page C-1.

B.3.1 Installing CA-IDMS/VSAM Transparency in the CICS environment

Initial Installation: When installing CA-IDMS/VSAM Transparency in the CICS environment, a CICSOPTS module will be assembled and link edited as part of module IDMSINTC. All parameters for CICSOPTS that are required for the VSAM Transparency will be automatically generated by the CAIJMP installation utility when you indicate the product is to be installed, either as part of an integrated solution or as a single product during ADDON install.

Modifying the CICSOPTS: If you need to reassemble CICSOPTS to change any installation options, you must edit the CICSOPTS member, re-assemble your CICSOPTS module and linkedit IDMSINTC. Job control to do this should be taken from the job control that was generated by CAIJMP for your initial base tape installation.: ►►For more information on the CICSOPT macro and its parameters, refer to the *CA-IDMS System Operations*

Important: Be sure PLT entries are created to execute IDMSINTC at CICS startup and shutdown. Additionally, be sure that PPT entries for CA-IDMS modules have been added to your CICS system. These are used by IDMSINTC in setting up the CA-IDMS environment.

Files to be accessed through VSAM/T must be defined through the FCT assemblies to be initially closed, enabled, so that CICS does not attempt to open them until IDMSINTC has been started.

In order for VSAM/T to work when IDMSINTC is defined in the PLT to be executed at startup, CA-IDMS CV must be brought up before CICS is brought up. If CICS is to be brought up before CA-IDMS CV is brought up, IDMSINTC must be started through a transaction code. Before starting IDMSINTC, take measures to prevent applications from attempting to access the VSAM/T files so that you can avoid "open" errors.

►►For information on adding PPT entries, see the CA-IDMS installation guide for your operating environment.

Appendix C. Variable-Length Record Considerations

C.1 About this appendix	C-3
C.2 Using variable-length CA-IDMS/VSAM Transparency records	C-4

C.1 About this appendix

This appendix contains the procedure you need to follow when using variable-length CA-IDMS/VSAM Transparency records.

C.2 Using variable-length CA-IDMS/VSAM Transparency records

You must follow these steps to use a variable-length CA-IDMS/VSAM Transparency record:

1. **Define the VSAM data structure to the CA-IDMS/DB schema.** In general, you define this data structure in the same way that you define a VSAM data structure that does not contain variable-length records.

Additionally, you should do the following when you define the VSAM data structure to the CA-IDMS/DB schema:

- **Define the records as variable.** You do this by using the MINIMUM ROOT LENGTH and MINIMUM FRAGMENT LENGTH clauses of the schema ADD RECORD Data Description Language (DDL) statement.
- **Specify that the CA-IDMS/VSAM Transparency variable-length record procedure (ESVSPVLR) will be called at runtime.** You do this by using the CALL clause of the ADD RECORD statement, as follows:

```
CALL ESVSPVLR AFTER GET  
CALL ESVSPVLR BEFORE STORE  
CALL ESVSPVLR BEFORE MODIFY
```

►► For a complete description of the ADD RECORD statement, refer to *CA-IDMS Database Administration*.

2. **Specify the maximum length of the variable-length record to the FMT compiler.** You specify this value by using the VARIABLE UP TO option of the VSAM RECORD LENGTH IS statement. The record length can be obtained from the MAXLRECL field of the corresponding VSAM LISTCAT.

►► For information on FMT syntax, refer to Chapter 4, “Preparing Control Information” on page 4-1.

3. **Include an ADD PROGRAM statement for ESVSPVLR** in the DC/UCF system definition of all DC/UCF systems to be used with CA-IDMS/VSAM Transparency. Specify the ADD PROGRAM statement as follows:

```
ADD PROGRAM ESVSPVLR  
LANGUAGE IS ASSEMBLER  
REENTRANT  
RESIDENT.
```

Note: As part of the normal installation process for CA-IDMS/VSAM Transparency, your system definition will be updated with the definition of all programs required for this product. If you need to add these definitions to another system, source module DLODEVSM in your installed library contains the definition of all the CA-IDMS/VSAM Transparency programs.

Appendix D. CA-IDMS/VSAM Transparency Return Codes and Messages

D.1 About this appendix	D-3
D.2 Compiler messages	D-4
D.2.1 FMT compiler messages	D-4
D.2.2 TNT Compiler messages	D-7
D.3 Command interface messages	D-11
D.3.1 Command interface messages — OS/390	D-11
D.3.2 Command interface messages — VSE/ESA	D-13
D.3.3 Front-end messages — OS/390 and VSE/ESA	D-16
D.4 Run-time messages — OS/390	D-17
D.5 Run-time messages — VSE/ESA	D-18
D.6 Migration messages	D-19
D.6.1 Migration messages — OS/390	D-19
D.6.2 Migration messages — VSE/ESA	D-19
D.7 Run-time feedback codes	D-20
D.7.1 OPEN codes	D-20
D.7.2 CLOSE/TCLOSE codes	D-22
D.7.3 REQUEST macro codes	D-23

D.1 About this appendix

This appendix contains the return codes and messages issued by CA-IDMS/VSAM Transparency and are presented as follows:

- Compiler messages, which include:
 - File management table compiler messages
 - Transaction name table compiler messages
- Command interface messages
- Run-time messages for OS/390 and VSE/ESA
- Run-time feedback codes

D.2 Compiler messages

For both FMT and TNT compiler messages, severity codes indicate the significance of the error. The following severity codes apply:

- **W** (Warning) alerts the user to potential problems; processing continues.
- **E** (Error) indicates a nonfatal error; processing continues.
- **F** (Fatal) indicates a fatal error; processing terminates.

FMT and TNT compiler messages are presented separately below.

D.2.1 FMT compiler messages

The FMT compiler can issue the following return codes and messages:

Return code	Message
225001	'EJECT' NOT ALONE ON CARD. TOKEN ASSUMED EJECT must appear as the only entry on the card, unless the function of the card is to be other than that of a compiler directive. Severity: W
225002	INVALID 'SPACE' COMMAND PARAMETER SPACE must be followed by a blank and, optionally, a one-digit number greater than zero indicating the number of lines to be spaced. Severity: W
225003	'SPACE' NOT ALONE ON CARD. TOKEN ASSUMED SPACE must appear as the only entry on the card, unless the function of the card is to be other than that of a compiler directive. Severity: W
225004	SEQUENCE ERROR. RUN ABORTED. Statements must be in sequential order. Severity: F
225005	STRING EXCEEDS MAXIMUM OR AVAILABLE LENGTH The number of characters in the character string specified exceeds the length that is allowed for this particular parameter. Severity: E

Return code	Message
225006	HEX STRING EXCEEDS MAXIMUM OR AVAILABLE LENGTH The number of characters in the hexadecimal string specified exceeds the length that is allowed for this particular parameter. Severity: E
225007	HEX STRING CONTAINS INVALID CHARACTERS Nonhexadecimal characters appear in the hexadecimal string specified. Severity: E
225008	DIAGNOSTIC TABLE SIZE EXCEEDED. TOO MANY ERRORS. Too many errors have occurred, causing an overflow of the diagnostic table. Correct the previous error. Severity: F
225009	INVALID ICTL PARAMETER SPECIFICATION The input control parameter was specified incorrectly. Check the syntax. Severity: E
225010	INVALID OCTL PARAMETER SPECIFICATION The output control parameter was specified incorrectly. Check the syntax. Severity: E
225011	INVALID ISEQ PARAMETER SPECIFICATION The input sequence parameter was specified incorrectly. Check the syntax. Severity: E
225012	UNEXPECTED END OF FILE (PERIOD MISSING) Invalid syntax has been encountered. Check for missing periods. Severity: F
225013	INVALID KEYWORD, SKIPPING TO NEXT STATEMENT The keyword encountered is not valid. Compilation resumes with the next statement. Severity: E

Return code	Message
225014	EXPECTED PERIOD, SKIPPING TO NEXT STATEMENT A period is missing after a statement. Statements must end with a period. Severity: E
225015	INVALID FMT NAME An FMT name must be specified as a 1- to 8-character alphanumeric string. Severity: E
225016	INVALID FILE TYPE File type must be KSDS/PATH/ESDS/RRDS. Severity: F
225017	INVALID NUMBER SPECIFIED Buffer space must be specified as a 1- to 5-digit number that ranges from 0 through 32,767. Severity: E
225018	INVALID STARTING POSITION Key or relative-record-number position must be specified as a 1- to 5-digit number that ranges from 0 through 32,767. Severity: E
225019	INVALID LENGTH SPECIFICATION Key length must be specified as a 1- to 3-digit number that ranges from 1 to 255. Severity: E
225020	INVALID EXIT NAME An exit name must be specified as a 1- to 8-character alphanumeric string. Severity: E
225021	INVALID RECORD NAME A record name must be specified as a 1- to 16-character alphanumeric string. Severity: E

Return code	Message
225022	<p>INVALID SET NAME</p> <p>A set name must be specified as a 1- to 16-character alphanumeric string.</p> <p>Severity: E</p>
225023	<p>MISSING KEY LENGTH FOR KSDS OR PATH</p> <p>A key length statement is required for file type KSDS or PATH.</p> <p>Severity: E</p>
225024	<p>MISSING KEY POSITION FOR KSDS OR PATH</p> <p>A key position statement is required for file type KSDS or PATH.</p> <p>Severity: E</p>
225025	<p>KEY LENGTH INVALID FOR ESDS OR RRDS</p> <p>A key length statement is not allowed for file type ESDS or RRDS.</p> <p>Severity: E</p>
225026	<p>KEY POSITION INVALID FOR ESDS OR RRDS</p> <p>A key position statement is not allowed for file type ESDS or RRDS.</p> <p>Severity: E</p>
225027	<p>RRN POSITION INVALID FOR KSDS, PATH, ESDS</p> <p>A relative-record-number position statement is used for file type RRDS only.</p> <p>Severity: E</p>

D.2.2 TNT Compiler messages

The TNT compiler can issue the following return codes and messages:

Return code	Message
226001	<p>'EJECT' NOT ALONE ON CARD. TOKEN ASSUMED</p> <p>EJECT must appear as the only entry on the card, unless the function of the card is to be other than that of a compiler directive.</p> <p>Severity: W</p>

Return code	Message
226002	INVALID 'SPACE' COMMAND PARAMETER SPACE must be followed by a blank and, optionally, a one-digit number greater than zero that indicates the number of lines to be spaced. Severity: W
226003	'SPACE' NOT ALONE ON CARD. TOKEN ASSUMED SPACE must appear as the only entry on the card, unless the function of the card is to be other than that of a compiler directive. Severity: W
226004	SEQUENCE ERROR. RUN ABORTED. Statements must be in sequential order. Severity: F
226005	STRING EXCEEDS MAXIMUM OR AVAILABLE LENGTH The number of characters in the character string specified exceeds the length that is allowed for this particular parameter. Severity: E
226006	HEX STRING EXCEEDS MAXIMUM OR AVAILABLE LENGTH The number of characters in the hexadecimal string specified exceeds the length that is allowed for this particular parameter. Severity: E
226007	HEX STRING CONTAINS INVALID CHARACTERS Nonhexadecimal characters appear in the hexadecimal string specified. Severity: E
226008	DIAGNOSTIC TABLE SIZE EXCEEDED. TOO MANY ERRORS Too many errors have occurred, causing an overflow of the diagnostic table. Correct the previous error. Severity: F
226009	INVALID ICTL PARAMETER SPECIFICATION The input control parameter was specified incorrectly. Check the syntax. Severity: E

Return code	Message
226010	INVALID OCTL PARAMETER SPECIFICATION The output control parameter was specified incorrectly. Check the syntax. Severity: E
226011	INVALID ISEQ PARAMETER SPECIFICATION The input sequence parameter was specified incorrectly. Check the syntax. Severity: E
226012	INVALID CORE SIZE PARAMETER SPECIFICATION Severity: E
226013	UNEXPECTED END OF FILE (PERIOD MISSING) Invalid syntax has been encountered. Check for missing periods. Severity: F
226014	INVALID KEYWORD, SKIPPING TO NEXT STATEMENT The keyword encountered is not valid. Compilation resumes with the next statement. Severity: E
226015	EXPECTED PERIOD, SKIPPING TO NEXT STATEMENT A period is missing after a statement. Statements must end with a period. Severity: E
226016	INVALID TNT NAME SPECIFICATION A TNT name must be a 1- to 8-character alphanumeric string. Severity: E
226017	INVALID SUBSCHEMA NAME SPECIFICATION The subschema was specified incorrectly. Check the syntax. Severity: E
226018	INTERNAL WORK AREA SIZE EXCEEDED The work area is too small. Check the buffer size specification. Severity: F
226019	DUPLICATE TASK NAME The task name must be unique. Severity: E

Return code	Message
226020	INVALID TASK NAME SPECIFICATION The task name was specified incorrectly. Check the syntax. Severity: E

D.3 Command interface messages

The CA-IDMS/VSAM Transparency command interface messages for OS/390 and VSE/ESA are described below.

D.3.1 Command interface messages — OS/390

The CA-IDMS/VSAM Transparency command interface can issue the following messages for OS/390:

Message ID	Message
ES227001	VSAM/T ACTIVE, ENTER REPLY The CA-IDMS/VSAM Transparency subsystem is initialized and ready to accept requests.
ES227002	VSAM/T INACTIVE, ENTER REPLY The CA-IDMS/VSAM Transparency subsystem is not active and will not accept any requests.
ES227003	VSAM/T QUIESCING, ENTER REPLY The CA-IDMS/VSAM Transparency subsystem is shutting down. All jobs currently using CA-IDMS/VSAM Transparency will be allowed to continue. No requests from new jobs will be accepted.
ES227004	VSAM/T ABORTED, ENTER REPLY The CA-IDMS/VSAM Transparency subsystem has performed an immediate shutdown. No requests from new jobs or jobs in progress will be accepted.
ES227005	VSAM/T INVALID REPLY The last reply to the CA-IDMS/VSAM Transparency command interface program was not a valid CA-IDMS/VSAM Transparency command. The current status of the CA-IDMS/VSAM Transparency subsystem is redisplayed, along with a request to reply with another command.
ES227006	VSAM/T STATUS ERROR, ENTER REPLY The CA-IDMS/VSAM Transparency command interface is not able to determine the current status of the CA-IDMS/VSAM Transparency subsystem. This could be due to a system internal error.
ES227007	VSAM/T INITIALIZATION STARTING The CA-IDMS/VSAM Transparency command interface is starting the CA-IDMS/VSAM Transparency subsystem initialization process.

Message ID	Message
ES227008	VSAM/T SSCT NOT FOUND The CA-IDMS/VSAM Transparency command interface cannot locate the subsystem communications table. Check that the CA-IDMS/VSAM Transparency subsystem name has been specified in SYS1.PARMLIB(IEFSSNxx) and that the operating system has been cycled since the subsystem name was added.
ES227009	VSAM/T SSSM BLDL FAILED The CA-IDMS/VSAM Transparency command interface cannot locate the CA-IDMS/VSAM Transparency subsystem manager in the authorized load library. Check the startup procedure (ESVS) in SYS1.PROCLIB to ensure that the correct authorized load library is specified for APFLIB and that the subsystem manager load module (ESVSSSSM) is in the authorized library.
ES227010	VSAM/T SSSM LOAD FAILED The CA-IDMS/VSAM Transparency command interface cannot load the CA-IDMS/VSAM Transparency subsystem manager (SSSM) from the authorized load library. Check that the subsystem manager load module (ESVSSSSM) has been linked or copied correctly.
ES227011	VSAM/T GETMAIN FAILED The CA-IDMS/VSAM Transparency command interface request to acquire storage in the OS/390 common system area (CSA) for the CA-IDMS/VSAM Transparency address space vector table has failed.
ES227012	VSAM/T INVALID EAVT ENCOUNTERED The CA-IDMS/VSAM Transparency address space vector table has been corrupted.
ES227013	VSAM/T CBMMEP1 NOT FOUND System internal error.
ES227014	VSAM/T CBMMEP2 NOT FOUND System internal error.
ES227015	VSAM/T INVALID SSVT ENCOUNTERED System internal error.
ES227016	VSAM/T INVALID SSSM NAME System internal error.

Message ID	Message
ES227017	<p>VSAM/T ESVSINIT ALREADY RUNNING</p> <p>An attempt was made to start the CA-IDMS/VSAM Transparency command interface program while it was already running. Use the command interface to enter CA-IDMS/VSAM Transparency commands.</p>
ES227018	<p>VSAM/T <i>nn</i> ACTIVE JOB(S)</p> <p>The CA-IDMS/VSAM Transparency command interface displays this message in response to the DISPLAY command. If there are active jobs using CA-IDMS/VSAM Transparency, <i>nn</i> will be the number of active jobs. A list of active jobs will follow the message. If there are no active jobs using CA-IDMS/VSAM Transparency, <i>nn</i> will be 00.</p>
ES227019	<p>VSAM/T EAVTVCBM IS EMPTY</p> <p>The pointer on the CA-IDMS/VSAM Transparency vector table to the VSAM control block manager has not been filled in.</p>

D.3.2 Command interface messages — VSE/ESA

The CA-IDMS/VSAM Transparency command interface messages that are issued in the VSE/ESA environment are presented below.

Note: If the command interface program is reading its input from the operator console (in response to a CONSOLE command), the suffix ENTER REPLY will appear in some of the messages.

Message ID	Message
ES227101	<p>VSAM/T ACTIVE</p> <p>VSAM/T ACTIVE, ENTER REPLY</p> <p>The CA-IDMS/VSAM Transparency system is initialized and ready to accept requests.</p>
ES227102	<p>VSAM/T INACTIVE</p> <p>VSAM/T INACTIVE, ENTER REPLY</p> <p>The CA-IDMS/VSAM Transparency system is not active and will not accept any requests.</p>
ES227103	<p>VSAM/T QUIESCING</p> <p>VSAM/T QUIESCING, ENTER REPLY</p> <p>The CA-IDMS/VSAM Transparency system is shutting down. All jobs currently using CA-IDMS/VSAM Transparency will be allowed to continue. No requests from new jobs will be accepted.</p>

Message ID	Message
ES227104	VSAM/T ABORTED VSAM/T ABORTED, ENTER REPLY <p>The CA-IDMS/VSAM Transparency system has performed an immediate shutdown. No requests from new jobs or jobs in progress will be accepted.</p>
ES227105	VSAM/T INVALID INPUT VSAM/T INVALID REPLY <p>The last reply to the CA-IDMS/VSAM Transparency command interface program was not a valid CA-IDMS/VSAM Transparency command. If the command interface is reading its input from SYSIPT, it ignores the invalid command and reads the next command. If the command interface is reading its input from the operator console, it displays the current status of the CA-IDMS/VSAM Transparency system, along with a request to reply with another command.</p>
ES227106	VSAM/T STATUS ERROR VSAM/T STATUS ERROR, ENTER REPLY <p>The CA-IDMS/VSAM Transparency command interface is not able to determine the current status of the CA-IDMS/VSAM Transparency subsystem. This could be due to a system internal error.</p>
ES227107	VSAM/T INITIALIZATION STARTING <p>The CA-IDMS/VSAM Transparency command interface is starting the CA-IDMS/VSAM Transparency system initialization process.</p>
ES227108	CA-IDMS/VSAM Transparency SPVT NOT FOUND <p>Either the CA-IDMS/VSAM Transparency command interface program could not load the CA-IDMS/VSAM Transparency SVA phase vector table (SPVT), or the SPVT phase was loaded but is not in the OS/390 shared virtual area (SVA).</p>
ES227109	VSAM/T DOSM NOT FOUND <p>The CA-IDMS/VSAM Transparency command interface cannot locate the CA-IDMS/VSAM Transparency system manager (DOSM) in a core-image library. Check the JCL used to run the command interface program to ensure that the correct core-image library has been specified and is included in the LIBDEF search chain. Also check that the CA-IDMS/VSAM Transparency system manager phase (ESVSDOSM) is in the core-image library.</p>

Message ID	Message
ES227110	CA-IDMS/VSAM Transparency DOSM LOAD FAILED The CA-IDMS/VSAM Transparency command interface cannot load the CA-IDMS/VSAM Transparency system manager from the core-image library. Check that the system manager phase (ESVSDOSM) has been linked or copied correctly.
ES227111	VSAM/T EAVT GETVIS FAILED The CA-IDMS/VSAM Transparency command interface request to acquire storage in the OS/390 system GETVIS area for the CA-IDMS/VSAM Transparency active partition vector table has failed.
ES227112	VSAM/T INVALID EAVT ENCOUNTERED The CA-IDMS/VSAM Transparency active partition vector table has been corrupted.
ES227113	VSAM/T SYSIPT/SYSLST ERROR, CODE: <i>nnnn</i> The CA-IDMS/VSAM Transparency command interface encountered an error on SYSIPT or SYSLST. The codes are as follows: <ul style="list-style-type: none">■ 2100 -- Could not open SYSIPT■ 2101 -- Could not open SYSLST■ 2102 -- Error reading SYSIPT■ 2103 -- Error writing SYSLST
ES227114	VSAM/T SVC FAILED The CA-IDMS/VSAM Transparency command interface received a nonzero return code from the IDMS SVC. The SVC number used is the one specified in the CA-IDMS/VSAM Transparency SVA phase vector table (SPVT). Check the following: <ul style="list-style-type: none">■ That the correct SVC number has been specified in the SPVT■ That IDMSSEVC was run once after the operating system was cycled and before the CA-IDMS/VSAM Transparency command interface program was run
ES227115	VSAM/T PFIX FAILED System internal error.

Message ID	Message
ES227117	<p>VSAM/T ESVSINIT ALREADY RUNNING</p> <p>An attempt was made to start the CA-IDMS/VSAM Transparency command interface program while it was already running. If the command interface is reading its input from SYSIPT, wait for it to finish. If the command interface is reading its input from the operator console, use the interface to enter CA-IDMS/VSAM Transparency commands.</p>
ES227118	<p>VSAM/T <i>nn</i> ACTIVE JOB(S)</p> <p>The CA-IDMS/VSAM Transparency command interface displays this message in response to the DISPLAY command. If there are active jobs using CA-IDMS/VSAM Transparency, <i>nn</i> will be the number of active jobs. A list of active jobs will follow the message. If there are no active jobs using CA-IDMS/VSAM Transparency, <i>nn</i> will be 00.</p>
ES227119	<p>VSAM/T CA-IDMS SVC <i>nnn</i> ERROR</p>
ES227120	<p>VSAM/T CA-SYSTEM/ADAPTER ERROR</p>

D.3.3 Front-end messages — OS/390 and VSE/ESA

The CA-IDMS/VSAM Transparency front-end can write the following messages to the job log or console, depending on the MSG parameter value.

Message ID	Message
ES227401	<p>VSAM/T FILE <i>DDname/DLBL name</i> RPL <i>RPL-address</i> FDBK=<i>feedback-code</i></p> <p>The message gives the DDname or DLBL name, the address of the RPL, the IDMS status, and the feedback code.</p>
ES227410	<p>EXIT LIST module name not supported</p> <p>The ACB exit list contains a module name where the address of a module is expected to be.</p>

D.4 Run-time messages — OS/390

Message ID	Message
ES227200	VSAM/T FILE <i>filename</i> FUNCTN Indicates successful open or close.
ES227201	VSAM/T FILE <i>filename</i> FUNCTN ERROR FDBK = Indicates error occurred during open or close.
ES227202	VSAM/T INVALID PARM Error found when reading DD statement.
ES227203	VSAM/T DUPLICATE PARM Error found when reading DD statement.
ES227204	VSAM/T MISSING PARM Error found when reading DD statement.
ES227205	VSAM/T IS NOT ACTIVE VSAM/T must be initiated before application is started.
ES227206	VSAM/T FOUND INVALID EAVT System internal error.
ES227207	VSAM/T FOUND INVALID EJCT System internal error.
ES227208	VSAM/T GETMAIN FAILED System internal error.
ES227210	VSAM/T CONVERT/INTERPRET FAILED System internal error.
ES227211	VSAM/T ALLOCATION FAILED Rerun the job. A prior VSAM/T job of the same name failed, probably with a JCL error. But the failure of this job reset the subsystem control information; so when you rerun this job, it will execute successfully.

D.5 Run-time messages — VSE/ESA

Message ID	Message
ES227300	VSAM/T FILE <i>filename</i> FUNCTION Indicates successful open or close.
ES227301	VSAM/T FILE <i>filename</i> FUNCTN ERROR FDBK = Indicates error occurred during open or close.
ES227302	VSAM/T INVALID PARM Error occurred while reading the DLBL statement.
ES227303	VSAM/T DUPLICATE PARM Error occurred while reading the DLBL statement.
ES227304	VSAM/T MISSING PARM Error occurred while reading the DLBL statement.
ES227305	VSAM/T IS NOT ACTIVE VSAM/T must be initiated before application can be started.
ES227306	VSAM/T FOUND INVALID EAVT System internal error.
ES227307	VSAM/T FOUND INVALID EJCT System internal error.
ES227308	VSAM/T EJCT GETVIS FAILED System internal error.
ES227309	VSAM/T FOUND INVALID EDWA System internal error.
ES227310	VSAM/T LOCK GETVIS FAILED System internal error.
ES227311	VSAM/T LOCK APPLY FAILED System internal error.
ES227312	VSAM/T PARM GETVIS FAILED System internal error.
ES227313	VSAM/T EDWA GETVIS FAILED System internal error.

D.6 Migration messages

D.6.1 Migration messages — OS/390

Message ID	Message
ES229001	VSAM/T INPUT FILE OPEN FAILED
ES229002	VSAM/T OUTPUT FILE OPEN FAILED
ES229003	VSAM/T GETMAIN FAILED
ES229004	VSAM/T GET FAILED
ES229005	VSAM/T PUT FAILED FDBK CODE IS
ES229006	VSAM/T INPUT FILE CLOSE FAILED
ES229007	VSAM/T OUTPUT FILE CLOSE FAILED

D.6.2 Migration messages — VSE/ESA

Message ID	Message
ES229101	VSAM/T INPUT FILE OPEN FAILED
ES229102	VSAM/T OUTPUT FILE OPEN FAILED
ES229103	VSAM/T GETMAIN FAILED
ES229104	VSAM/T GET FAILED
ES229105	VSAM/T PUT FAILED
ES229106	VSAM/T INPUT FILE CLOSE FAILED
ES229107	VSAM/T OUTPUT FILE CLOSE FAILED

D.7 Run-time feedback codes

The following list describes all messages issued by CA-IDMS/VSAM Transparency at run time. The first message in each group is the VSAM message in the following format:

- The first two bytes are the contents of R15.
- The second two bytes are the ACB feedback code.

CA-IDMS/VSAM Transparency issues additional explanatory messages to document the error. Run-time messages are issued for:

- OPEN
- CLOSE/TCLOSE
- REQUEST macros

D.7.1 OPEN codes

The following table lists the codes issued for the OPEN command.

Feedback code	Description
X'0804'	THIS ACB IS ALREADY OPEN
X'01080004'	ACB IS ALREADY OPEN - AMFE
X'01080008'	ACB W/LOCAL SHARED RESOURCES - AMFE
X'0832'	VSAM PHASES CANNOT BE LOADED
X'01080032'	CAN'T LOAD TNT - AMBE
X'02080032'	CAN'T LOAD BEFORE EXIT - AMBE
X'03080032'	CAN'T LOAD AFTER EXIT _ AMBE
X'04080032'	BAD LINK CICS INTERFACE - AMFE
X'05080032'	FILE NOT DEFINED IN CICS FCT
X'0850'	VOLUME NOT AVAILABLE
X'01080050'	PREVIOUS FATAL ERROR - DOSM, SSSM OPEN
X'02080050'	PREVIOUS FATAL ERROR - DOSM BIND
X'03080050'	CAN'T BIND RUN-UNIT - AMBE
X'04080050'	CAN'T BIND REQ-UNIT - AMFE
X'086E'	INPUT FILE IS EMPTY
X'0108006E'	CAN'T OPEN FILE INPUT IF EMPTY-AMBE
X'0880'	DLBL STATEMENT MISSING OR INVALID

Feedback code	Description
X'01080080'	DLBL STATEMENT INVALID FMT NAME - DOSM
X'02080080'	DD STATEMENT INVALID SUBSYS PARMS - SSSM
X'0884'	PERMANENT I/O ERROR READING LABEL
X'01080084'	BAD EJCT PASSED TO AMFE - AMFE
X'02080084'	INVALID EJCT FUNCTION - AMFE
X'0888'	NOT ENOUGH STORAGE AVAILABLE
X'01080088'	CAN'T ACQUIRE EJCT, RWA, ETC. - AMBE
X'02080088'	CAN'T ACQUIRE EAMB - AMBE
X'03080088'	CAN'T ACQUIRE BUFFERS - AMFE
X'04080088'	CAN'T ACQUIRE EJCT, RWA, ETC - DOSM
X'05080088'	CAN'T ACQUIRE MORE EAMBS - DOSM
X'06080088'	GETVIS FOR LOCK FAILED - DOSM
X'07080088'	APPLY OF LOCK FAILED - DOSM
X'0890'	ERROR WHILE ACCESSING THE CATALOG
X'01080090'	BAD ACBID - AMFE, DOSM
X'02080090'	BAD SEND/RECEIVE STATUS - AMFE
X'03080090'	BAD EAMB OR EAMB POOL - AMFE
X'04080090'	ONLY BATCH CAN RUN LOCAL - AMBE
X'0894'	NO CATALOG ENTRY FOR THIS FILE
X'01080094'	CAN'T LOAD THE FMT - AMBE
X'08A0'	FILE TYPE VS. ACCESS MODE CONFLICT
X'010800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'020800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'030800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'040800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'050800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'060800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'070800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'080800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'090800A0'	FMT AND SUBSCHEMA CONFLICT - AMBE
X'08A4'	PERMANENT I/O ERROR READING VOLUME LABEL

Feedback code	Description
X'010800A4'	INVALID EAVT - SSSM, DOSM OPEN
X'020800A4'	INVALID EJCT - SSSM, DOSM OPEN
X'030800A4'	INVALID EDWA - DOSM
X'08A8'	FILE NOT AVAILABLE, ALREADY IN USE
X'010800A8'	DDNAME ALREADY IN USE - AMFE
X'08B4'	ERROR OPENING CATALOG
X'010800B4'	VSAM/T INACTIVE - DOSM, SSSM OPEN
X'020800B4'	VSAM/T INACTIVE - DOSM BIND
X'030800B4'	VSAM/T QUIESCING - DOSM OPEN
X'040800B4'	VSAM/T QUIESCING - DOSM BIND
X'050800B4'	VSAM/T INACTIVE - SSSM C/I
X'060800B4'	VSAM/T INACTIVE - SSSM ALLOC
X'070800B4'	VSAM/T QUIESCING - SSSM C/I
X'048800B4'	VSAM/T QUIESCING - SSSM ALLOC
X'08E8'	RST SPECIFIED FOR NON-REUSABLE FILE
X'010800E8'	OPEN RESET NOT ALLOWED - AMBE
X'020800E8'	OPEN RESET FAILED - AMBE
X'030800E8'	FMT HAS INVALID ERASE OPTION - AMBE
X'08FF'	UNEXPECTED ERROR PROCESSING CATALOG
X'010800FF'	CONTROL DLBL IN ERROR - DOSM OPEN
X'020800FF'	CONTROL DLBL IN ERROR - DOSM OPEN
X'030800FF'	CONTROL DLBL IN ERROR - DOSM OPEN

D.7.2 CLOSE/TCLOSE codes

The following table lists the codes issued for the CLOSE and TCLOSE commands.

R15 ACB feed-back	Description
X'0802'	FOUND INVALID CNTL BLOCKS THIS ACB
X'01080002'	CAN'T FIND ACB FOR EAMB - AMFE
X'0804'	ACB ALREADY CLOSED
X'01080004'	ACB ALREADY CLOSED - AMFE

R15 ACB feed-back	Description
X'0890'	ERROR WHILE ACCESSING THE CATALOG
X'01080090'	CLOSE SEND/RCV FAILED - AMFE
X'02080090'	CLOSE FINISH FAILED - AMFE
X'03080090'	VSAM/T INACTIVE - DOSM, SSSM CLOSE
X'04080090'	VSAM/T INACTIVE - DOSM FINISH
X'05080090'	VSAM/T INACTIVE - DOSM ACLOSE
X'06080090'	PREV. FATAL ERROR - DOSM, SSSM CLOSE
X'07080090'	PREVIOUS FATAL ERROR - DOSM FINISH
X'08080090'	PREVIOUS FATAL ERROR - DOSM ACLOSE
X'08E4'	INVALID VRP FOUND DURING CLOSE
X'010800E4'	INVALID EAVT - DOSM, SSSM CLOSE
X'020800E4'	INVALID EJCT - DOSM, SSSM CLOSE
X'030800E4'	INVALID EDWA - DOSM CLOSE
X'040800E4'	JOB ROLLED BACK OR DB LOCKED - SSSM
X'050800E4'	ACTIVE TRANSACTS. ROLLED BACK - SSSM

D.7.3 REQUEST macro codes

The following table lists the codes issued for the REQUEST macro.

►► CA-IDMS-DB abend codes and minor codes are fully documented in *CA-IDMS Messages and Codes*.

RPL feedback	Description
X'00000008'	NON-UNIQUE KEY IN ALT. INDEX
X'01000008'	DUPL KEY FOUND IN PATH - DKCK
X'00080004'	END OF FILE
X'01080004'	KEY > LARGEST, DIRECT - PRPL
X'02080004'	END OF FILE, SEQ. - PRPL
X'03080004'	END OF FILE, SEQ. - PRPL
X'04080004'	KEY > LARGEST, POINT - PRPL
X'00080008'	DUPLICATE KEY ERROR

RPL feedback	Description
X'01080008'	IDMS ERROR STATUS 0705 - duplicates violation
X'02080008'	IDMS ERROR STATUS 0805 - duplicates violation
X'03080008'	IDMS ERROR STATUS 1205 - duplicates violation
X'00080010'	RECORD NOT FOUND
X'01080010'	RECORD NOT FOUND, DIRECT - PRPL
X'02080010'	RECORD NOT FOUND, POINT - PRPL
X'03080010'	RECORD NOT FOUND, GENERIC GET -PRPL
X'04080010'	RECORD NOT FOUND, GENERIC POINT-PRPL
X'00080018'	RECORD ON A VOLUME OR EXTENT NOT AVAIL.
X'01080018'	IDMS ERROR STATUS 0301 - area not readied
X'02080018'	IDMS ERROR STATUS 0370 - area will not ready
X'03080018'	IDMS ERROR STATUS 0970 - area will not ready
X'04080018'	IDMS ERROR STATUS 1470 - area will not ready
X'0008001C'	NO MORE SPACE AVAILABLE
X'0108001C'	IDMS ERROR STATUS 0811 - area full
X'0208001C'	IDMS ERROR STATUS 1211 - area full
X'00080020'	INVALID RBA HAS BEEN SPECIFIED
X'01080020'	IDMS ERROR STATUS 1201 - area not readied
X'02080020'	IDMS ERROR STATUS 1202 - invalid suggested dbkey
X'03080020'	IDMS ERROR STATUS 0302 - invalid dbkey
X'00080028'	NOT ENOUGH CONTIGUOUS MAIN STORAGE
X'01080028'	IDMS ERROR STATUS xx56 - insufficient main memory
X'02080028'	IDMS ERROR STATUS xx72 - insufficient main memory
X'03080028'	COULD NOT ACQUIRE AN EPHD - AMBE
X'0008002C'	AREALEN IN RPL LESS THAN RECORD LEN
X'0108002C'	AREALEN LESS THAN RECLEN OR 4 - AMFE
X'0208002C'	AREALEN LESS THAN RECLEN FOR VLR - PRPL
X'00080040'	NO AVAILABLE STRING
X'01080040'	NO AVAILABLE STRING - PRPL
X'02080040'	NO AVAILABLE STRING - GETEPLH
X'00080044'	TYPE OF ACCESS CONFLICT, ACB VS. RPL

RPL feedback	Description
X'01080044'	FMT VS. SUBS CONFLICT - AMBE GETEPHD
X'00080048'	KEYED ACCESS REQUESTED TO ESDS
X'01080048'	KEYED ACCESS TO ESDS, DIR -PRPL
X'02080048'	KEYED ACCESS TO ESDS, PNT -PRPL
X'0008005C'	PUT(UPD) OR ERASE W/O PRIOR GET(UPD)
X'0108005C'	PUT (UPD) W/O PRIOR GET(UPD) - PRPL
X'0208005C'	ERASE W/O PRIOR GET(UPD) - PRPL
X'00080068'	INVALID OR CONFLICTING RPL OPTIONS
X'01080068'	INVALID TYPE OF REQUEST - AMFE
X'02080068'	INVALID OR CONFLICT OPTIONS - AMFE
X'03080068'	LRD WITHOUT BWD - PRPL
X'04080068'	LRD WITHOUT BWD - PRPL
X'05080068'	KEY REQUIRED, BUT NOT SPECIFIED
X'0008006C'	INVALID RECLLEN SPECIFIED IN RPL
X'0108006C'	IDMS ERROR STATUS 0355 - invalid record length
X'0208006C'	IDMS ERROR STATUS 0555 - invalid record length
X'0308006C'	IDMS ERROR STATUS 0855 - invalid record length
X'0408006C'	IDMS ERROR STATUS 1255 - invalid record length
X'0508006C'	RPL RECLLEN > FMT RECLLEN
X'0008008C'	VSAM ENCOUNTERED BAD SPANNED RECORD
X'0108008C'	IDMS ERROR STATUS 0804
X'0208008C'	IDMS ERROR STATUS 1204
X'00080090'	AN INDEX POINTER ► AT NO BASE RECORD
X'01080090'	IDMS ERROR STATUS xx61 - invalid dbkey
X'00080098'	NO AVAILABLE BUFFER AT THIS TIME
X'01080098'	NO BUFFER FOR LOCATE MODE - AMFE
X'000800C0'	INVALID RELATIVE RECORD NUMBER
X'010800C0'	IDMS ERROR STATUS 0304
X'000800C8'	TRIED ADAC OR CNV THRU A PATH
X'010800C8'	ADAC OR CNV THRU A PATH - PRPL DIR
X'010800C8'	ADAC OR CNV THRU A PATH - PRPL PNT

RPL feedback	Description
X'000C0004'	READ DATA FAILED
X'010C0004'	SEND/RECEIVE ERROR, FE
X'020C0004'	P58 ERROR DETECTED, FE
X'030C0004'	FINISH ERROR, FE
X'040C0004'	STORAGE ERROR, FE
X'050C0004'	BIND ERROR, FE
X'060C0004'	BEFORE EXIT ERROR, PRPL
X'070C0004'	INTERNAL ERROR, EPLHACT PRPL
X'080C0004'	INTERNAL ERROR, RWAFUNC PRPL
X'090C0004'	AFTER EXIT ERROR, PRPL
X'0A0C0004'	INTERNAL ERROR, EPHDRCOD BCCK DIR
X'0B0C0004'	INTERNAL ERROR, NATV+RRNF DIRECT
X'0C0C0004'	INTERNAL ERROR, EPHDRCOD DIRECT
X'0E0C0004'	INTERNAL ERROR, EPHDRCOD BCCK SEQ
X'0D0C0004'	INTERNAL ERROR, EPHDRCOD SEQ.
X'0F0C0004'	INTERNAL ERROR, EPHDRCOD BCCK PUT
X'100C0004'	INTERNAL ERROR, EPHDRCOD PUT
X'110C0004'	INTERNAL ERROR, EPHDRCOD BCCK PNT
X'120C0004'	INTERNAL ERROR, NATV+RRNF PNT
X'130C0004'	INTERNAL ERROR, EPHDRCOD PNT
X'140C0004'	INTERNAL ERROR, EPHDRCOD BCCK ERASE
X'150C0004'	INTERNAL ERROR, INVALID ERASE OPTION
X'160C0004'	INTERNAL ERROR, EPHDRCOD ERASE
X'170C0004'	INTERNAL ERROR, BCCK NO SR51
X'180C0004'	INTERNAL ERROR, ERPLEPHD IS BAD
X'190C0004'	INTERNAL ERROR, EPLHBUFL IS BAD
X'1A0C0004'	N/A INTERNAL ERROR, EPLHBUFL IS BAD
X'1B0C0004'	INTERNAL ERROR, NO BIND ADDR (BCCK)
X'1C0C0004'	INTERNAL ERROR, P58S NO GOOD (SRCV)
X'1D0C0004'	INTERNAL ERROR, DKCK FROM BCCK
X'1E0C0004'	INTERNAL ERROR, DKCK FROM CIDMS

RPL feedback	Description
X'1F0C0004'	INTERNAL ERROR, RPL ► ACB NOT OPEN
X'200C0004'	INTERNAL ERROR, ACB NOT ► EAMB
X'210C0004'	INTERNAL ERROR, EAMB NOT ► EJCT
X'220C0004'	INTERNAL ERROR, CAN'T FIND CSA
X'230C0004'	INTERNAL ERROR, CAN'T FIND TCA
X'240C0004'	INTERNAL ERROR, INRC GETSTG FAILED
X'250C0004'	INTERNAL ERROR, EJCT NOT ► RWA
X'000C0008'	READ INDEX SET FAILED
X'010C0008'	IDMS ERROR STATUS 0208 - invalid record or set name
X'020C0008'	IDMS ERROR STATUS 0308 - invalid record or set name
X'030C0008'	IDMS ERROR STATUS 0508 - invalid record or set name
X'040C0008'	IDMS ERROR STATUS 0608 - invalid record or set name
X'050C0008'	IDMS ERROR STATUS 0708 - invalid record or set name
X'060C0008'	IDMS ERROR STATUS 1108 - invalid record or set name
X'070C0008'	IDMS ERROR STATUS 1208 - invalid record or set name
X'080C0008'	IDMS ERROR STATUS 1408
X'090C0008'	IDMS ERROR STATUS 1508
X'0A0C0008'	IDMS ERROR STATUS 1608
X'0B0C0008'	IDMS ERROR STATUS 2008
X'0C0C0008'	IDMS ERROR STATUS 0209
X'0D0C0008'	IDMS ERROR STATUS 0709
X'0E0C0008'	IDMS ERROR STATUS 0809
X'0F0C0008'	IDMS ERROR STATUS 1109
X'100C0008'	IDMS ERROR STATUS 1209
X'110C0008'	IDMS ERROR STATUS 0210
X'120C0008'	IDMS ERROR STATUS 0310
X'130C0008'	IDMS ERROR STATUS 0510
X'140C0008'	IDMS ERROR STATUS 0610
X'150C0008'	IDMS ERROR STATUS 0710
X'160C0008'	IDMS ERROR STATUS 0810
X'170C0008'	IDMS ERROR STATUS 0910

RPL feedback	Description
X'180C0008'	IDMS ERROR STATUS 1110
X'190C0008'	IDMS ERROR STATUS 1210
X'1A0C0008'	IDMS ERROR STATUS 2010
X'1B0C0008'	IDMS ERROR STATUS 1115
X'1C0C0008'	IDMS ERROR STATUS 0221
X'1D0C0008'	IDMS ERROR STATUS 0721
X'1E0C0008'	IDMS ERROR STATUS 0821
X'1F0C0008'	IDMS ERROR STATUS 1121
X'200C0008'	IDMS ERROR STATUS 1221
X'210C0008'	IDMS ERROR STATUS 0225
X'220C0008'	IDMS ERROR STATUS 0325
X'230C0008'	IDMS ERROR STATUS 1225
X'240C0008'	IDMS ERROR STATUS 1725
X'250C0008'	IDMS ERROR STATUS 0226
X'260C0008'	IDMS ERROR STATUS 0331
X'270C0008'	IDMS ERROR STATUS 0233
X'280C0008'	IDMS ERROR STATUS 0833
X'290C0008'	IDMS ERROR STATUS 1233
X'2A0C0008'	IDMS ERROR STATUS 0260
X'2B0C0008'	IDMS ERROR STATUS 0360
X'2C0C0008'	IDMS ERROR STATUS 0860
X'2D0C0008'	IDMS ERROR STATUS 1260
X'2E0C0008'	IDMS ERROR STATUS 1467
X'2F0C0008'	IDMS ERROR STATUS 0971
X'300C0008'	IDMS ERROR STATUS 1480
X'310C0008'	IDMS ERROR STATUS 1481
X'320C0008'	IDMS ERROR STATUS 1482
X'330C0008'	IDMS ERROR STATUS 1483
X'340C0008'	IDMS ERROR STATUS 1468
X'000C000C'	READ SEQUENCE SET FAILED
X'010C000C'	UNRECOGNIZED MAJOR/MINOR CODE

RPL feedback	Description
X'020C000C'	IDMS ERROR STATUS xx74 - load of module failed
X'030C000C'	IDMS ERROR STATUS xx75 - read error
X'040C000C'	IDMS ERROR STATUS xx76 - write error
X'050C000C'	IDMS ERROR STATUS xx77 - run-unit not bound
X'060C000C'	IDMS ERROR STATUS xx79 - too many dbkey locks
X'070C000C'	UNRECOGNIZED IDMS MINOR CODE
X'080C000C'	IDMS ERROR STATUS 0966 - area locked
X'000C0010'	WRITE DATA FAILED
X'010C0010'	IDMS ERROR STATUS xx60 - broken chain
X'020C0010'	IDMS ERROR STATUS xx65 - bad DMCL or JCL

Appendix E. CA-IDMS/VSAM Transparency User Exits

- E.1 About this appendix E-3
- E.2 COBOL user exit program E-4
 - E.2.1 Program requirements E-4
 - E.2.2 COBOL user exit control block E-5
 - E.2.3 Sample COBOL user exit E-11
- E.3 Assembler user exit program E-17
 - E.3.1 Program requirements E-17
 - E.3.2 Assembler user exit control block E-18
 - E.3.3 Assembler user exit template E-21
 - E.3.4 Sample assembler user exit E-22
 - E.3.5 Macros E-29

E.1 About this appendix

This appendix provides information on how to write user exit programs in COBOL and Assembler languages.

User exit programs: CA-IDMS/VSAM Transparency supports the use of user exit programs. You can define two exits for each FMT:

- **A before exit** is issued before CA-IDMS/VSAM Transparency processing begins or before the database is accessed. In a before exit, you can build the VSAM record layout from the CA-IDMS/DB structure.
- **An after exit** is issued after CA-IDMS/VSAM Transparency processing finishes or after the database is accessed. In an after exit, you can build the CA-IDMS/DB structure from the VSAM record layout.

You can write the exits in either COBOL or Assembler.

E.2 COBOL user exit program

When programming a COBOL user exit for CA-IDMS/VSAM Transparency, fulfill the requirements noted below. This discussion is followed by a description of the user exit control block, and a sample COBOL user exit.

E.2.1 Program requirements

Quasi-reentrant exit: The exit must be quasi-reentrant. Under central version, this requires no additional processing. Under local mode, the exit must initialize all work fields programmatically. A quasireentrant program retains the working storage values last set by a previous call to the program. CA-IDMS/DB and CA-IDMS/VSAM Transparency do not re-initialize the fields when executing in local mode.

ENVIRONMENT DIVISION: In the ENVIRONMENT DIVISION, specify MODE IS BATCH, even if the exit is to run under CA-IDMS/DC.

LINKAGE SECTION: In the LINKAGE SECTION, include:

- The user exit control block used to obtain information from CA-IDMS/VSAM Transparency that the user exit requires.
- The subschema control block that can be used by the user exit to perform its own DML calls for database access.
- The user record that the exit is building and storing. This is the record used by the VSAM program.

Example

```
LINKAGE SECTION.  
01 USER-EXIT-CONTROL-BLOCK.  
  .  
  .  
  .  
COPY IDMS SUBSCHEMA-CTRL.  
  .  
  .  
  .  
01 USER-RECORD.  
  .  
  .  
  .  
PROCEDURE DIVISION  
  USING USER-EXIT-CONTROL-BLOCK, SUBSCHEMA-CTRL, USER-RECORD.
```

PROCEDURE DIVISION: In the PROCEDURE DIVISION, do not copy SUBSCHEMA-BINDS. The rununit is bound prior to the exit receiving control. CA-IDMS/VSAM Transparency binds the record defined in the FMT the first time it attempts to access it. The exit must issue a bind only for additional records it needs.

- If the exit attempts to access the record defined in the FMT prior to CA-IDMS/VSAM Transparency, the exit must bind this record.

- The exit must issue a GOBACK when it is complete. It must not issue a STOP RUN.

►► For further information on coding CA-IDMS/DB and DC/UCF programs see *CA-IDMS Navigational DML Programming*.

Compiling and linking the program: Compile and link the program once the user exit is complete and you have run the program through the DMLC preprocessor. Link the following modules with the exit:

```
INCLUDE libname(ESVSIIDMS)
INCLUDE libname(IDMSBALI)
```

Note: Libname is the DD name of the file in the JCL that contains the CA-IDMS/DB-supplied object modules.

Defining the exit: Define the exit to CA-IDMS/DB by including an ADD PROGRAM statement for the exit program in the system generation. The ADD PROGRAM statement should appear as follows:

```
ADD PROGRAM exitname
    LANGUAGE IS COBOL
    QUASIREENTRANT
    NOPROTECT.
```

Note: The exit accesses DC/UCF system control blocks, therefore storage protection must be off (NOPROTECT).

E.2.2 COBOL user exit control block

The user exit control block contains fields that can be used by the user exit. Some fields can be modified, some can not. The COBOL code describing the record layout follows:

```
01 USER-EXIT-CONTROL-BLOCK.
05  EXB-ID                      PIC X(4).
05  EXB-USER                     PIC S9(8) COMP.
05  EXB-ADDR-RWA                 PIC S9(8) COMP.
05  EXB-ADDR-STACK              PIC S9(8) COMP.
05  EXB-ADDR-SSC                PIC S9(8) COMP.
05  FILLER                      PIC S9(8) COMP.
05  EXB-ADDR-RPL                PIC S9(8) COMP.
05  EXB-ADDR-ARGUMENT          PIC S9(8) COMP.
05  EXB-ADDR-RECORD-BUFFER     PIC S9(8) COMP.
05  EXB-MAX-KEY-LENGTH          PIC S9(4) COMP.
05  EXB-REC-BUFFER-LENGTH      PIC S9(4) COMP.
05  EXB-VSAM-REC-LENGTH        PIC S9(4) COMP.
05  EXB-VSAM-KEY-LENGTH        PIC S9(4) COMP.
05  EXB-VSAM-KEY-POSITION      PIC S9(4) COMP.
05  EXB-FMT-NAME                PIC X(8).
05  EXB-DD-NAME                 PIC X(8).
05  EXB-IDMS-REC-NAME          PIC X(16).
05  EXB-IDMS-SET-NAME          PIC X(16).
05  EXB-RESERVED                PIC S9(2) COMP.
05  EXB-FEEDBACK                PIC X(4).
05  EXB-RPL-REQUEST-TYPE       PIC X(6).
    88  GET-REQUEST              VALUE 'GET'.
    88  PUT-REQUEST              VALUE 'PUT'.
    88  POINT-REQUEST            VALUE 'POINT'.
    88  ENDREQ-REQUEST           VALUE 'ENDREQ'.
    88  ERASE-REQUEST            VALUE 'ERASE'.
05  EXB-RPL-OPTION-1.
    10  EXB-DIRECT                PIC X.
        88  DIRECT-REQUEST        VALUE 'X'.
    10  EXB-SEQUENTIAL           PIC X.
        88  SEQUENTIAL-REQUEST    VALUE 'X'.
    10  EXB-SKIP                 PIC X.
        88  SKIP-REQUEST          VALUE 'X'.
    10  EXB-KEY-GT-EQ            PIC X.
        88  KEY-GT-EQ-REQUEST     VALUE 'X'.
    10  EXB-GENERIC              PIC X.
        88  GENERIC-REQUEST       VALUE 'X'.
    10  FILLER                   PIC X(3).
05  EXB-RPL-OPTION-2.
    10  EXB-KEYED-ACCESS         PIC X.
        88  KEYED-ACCESS          VALUE 'X'.
    10  EXB-ADDRESS-ACCESS      PIC X.
        88  ADDRESS-ACCESS        VALUE 'X'.
    10  EXB-BACKWARD-ACCESS     PIC X.
        88  BACKWARD-ACCESS       VALUE 'X'.
    10  EXB-LAST-REC-ACCESS     PIC X.
        88  LAST-REC-ACCESS       VALUE 'X'.
```

```

10  EXB-UPDATE-ACCESS  PIC X.
    88  UPDATE-ACCESS  VALUE 'X'.
10  EXB-SET-POSITION  PIC X.
    88  SET-POSITION   VALUE 'X'.
10  FILLER             PIC X(2).
05  EXB-VSAM-FILE-TYPE.
10  EXB-KSDS          PIC X.
    88  KSDS           VALUE 'X'.
10  EXB-PATH          PIC X.
    88  PATH           VALUE 'X'.
10  EXB-RRDS          PIC X.
    88  RRDS           VALUE 'X'.
10  EXB-ESDS          PIC X.
    88  ESDS           VALUE 'X'.
10  EXB-BASE-CLUS-ESDS PIC X.
    88  BASE-CLUS-ESDS VALUE 'X'.
10  EXB-SET-DEFINED  PIC X.
    88  SET-DEFINED    VALUE 'X'.
10  EXB-NATIVE-VSAM  PIC X.
    88  NATIVE-VSAM    VALUE 'X'.
10  FILLER            PIC X.
05  EXB-EXIT-FLAGS.
10  EXB-EXIT-TYPE    PIC X.
    88  BEFORE-EXIT    VALUE 'B'.
    88  AFTER-EXIT     VALUE 'A'.
10  EXB-SKIP-TO-AFTER PIC X.
    88  SKIP-TO-AFTER  VALUE 'X'.
10  EXB-RETURN-IMMED PIC X.
    88  RETURN-IMMED   VALUE 'X'.
10  EXB-EXIT-SET-REC-LENGTH PIC X.
    88  EXIT-SET-REC-LENGTH VALUE 'X'.
10  EXB-EXIT-SET-ARGUMENT PIC X.
    88  EXIT-SET-ARGUMENT VALUE 'X'.
10  EXB-EXIT-SET-FEEDBACK PIC X.
    88  EXIT-SET-FEEDBACK VALUE 'X'.
10  FILLER            PIC X(2).
05  FILLER            PIC X(8).
05  FILLER            PIC X(2).
05  EXB-SAVE-AREA    PIC X(72).
05  FILLER            PIC X(12).

```

Field descriptions: The following table gives a detailed description of the fields in the exit control block.

Note: In the Type column, A means alphanumeric and N means numeric (COMP).

Offset	Name	Type	Len	Description
0	EXB-ID	A	4	Contains the literal 'EXB'.
4	EXB-USER	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.

Offset	Name	Type	Len	Description
8	EXB-ADDR-RWA	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
12	EXB-ADDR-STACK	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
16	EXB-ADDR-SSC	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
20	FILLER	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
24	EXB-ADDR-RPL	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
28	EXB-ADDR- ARGU- MENT	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
32	EXB-ADDR-RECORD -BUFFER	N	4	Can not be used by the COBOL exit. Reserved for the Assembler exit.
36	EXB-MAX-KEY- LENGTH	N	2	Maximum key length.
38	EXB-REC-BUFFER- LENGTH	N	2	Current length of record in the record buffer.
40	EXB-VSAM-REC- LENGTH	N	2	Contains the VSAM record length obtained from FMT.
42	EXB-VSAM-KEY- LENGTH	N	2	Contains the VSAM key length obtained from FMT.
44	EXB-VSAM-KEY- POSITION	N	2	Contains the VSAM key position obtained from FMT.
46	EXB-FMT-NAME	A	8	FMT name.
54	EXB-DD-NAME	A	8	DD name.
62	EXB-IDMS-REC-NAME	A	16	CA-IDMS/DB record name.
78	EXB-IDMS-SET-NAME	A	16	CA-IDMS/DB set name.

Offset	Name	Type	Len	Description
94	EXB-RESERVED	N	2	Reserved field.
96	EXB-FEEDBACK	A	4	Allows you to set a feedback code (return code) from the exit program. By placing a value here (such as subschema control status) and by moving an 'X' to EXIT-SET-FEEDBACK, CA-IDMS/VSAM Transparency will display an error message on the job log including the feedback code you set: <ul style="list-style-type: none"> ▪ Internal error id ▪ Return code ▪ Component code ▪ Error code
100	EXB-RPL-REQUEST	A	6	Identifies the type of VSAM call issued. Use this field in conjunction with EXB-EXIT-FLAGS to determine what type of processing will be required. Valid values are:
100	GET-REQUEST	A	6	'GET '
100	PUT-REQUEST	A	6	'PUT '
100	POINT-REQUEST	A	6	'POINT '
100	ENDREQ-REQUEST	A	6	'ENDREQ'
100	ERASE-REQUEST	A	6	'ERASE '
106	EXB-RPL-OPTION-1	A	8	
106	EXB-RPL-OPTION-1	A	8	Further qualifies the request type:
106	DIRECT-REQUEST	A	1	Direct request
107	SEQUENTIAL-REQUEST	A	1	Sequential request

Offset	Name	Type	Len	Description
108	SKIP-REQUEST	A	1	Skip request
109	KEY-GT-EQ-REQUEST	A	1	Key gt/eq request
110	GENERIC-REQUEST	A	1	Generic request
111	FILLER	A	3	Filler
114	EXB-RPL-OPTION-2	A	8	Further qualifies the request type:
114	KEYED-ACCESS	A	1	Keyed access
115	ADDRESS-ACCESS	A	1	Address access
116	BACKWARD-ACCESS	A	1	Backward access
117	LAST-REC-ACCESS	A	1	Last record access
118	UPDATE-ACCESS	A	1	Update access
119	SET-POSITION	A	1	Set position
120	FILLER	A	2	Filler
122	EXB-VSAM-FILE-TYPE	A	8	Identifies the VSAM file type:
122	KSDS	A	1	KSDS
123	PATH	A	1	PATH
124	RRDS	A	1	RRDS
125	ESDS	A	1	ESDS
126	BASE-CLUS-ESDS	A	1	Base cluster ESDS
127	SET-DEFINED	A	1	Set defined
128	NATIVE-VSAM	A	1	Native VSAM
129	FILLER	A	1	Filler
130	EXB-EXIT-FLAGS	A	8	Exit flags that control CA-IDMS/VSAM Transparency processing
130	EXB-EXIT-TYPE	A	1	Indicates the exit type:
130	BEFORE-EXIT	A	1	Before = 'B'
130	AFTER-EXIT	A	1	After = 'A'

The next 5 bytes describe exit processing. Put an X in the byte to turn the character switch on. The default is blank.

Offset	Name	Type	Len	Description
131	SKIP-TO-AFTER	A	1	Tells CA-IDMS/VSAM Transparency to invoke the after exit immediately after the before exit and bypass CA-IDMS/VSAM Transparency processing.
132	RETURN-IMMED	A	1	Tells CA-IDMS/VSAM Transparency to return immediately to the user and bypass CA-IDMS/VSAM Transparency processing.
133	EXIT-SET-REC-LENGTH	A	1	Tells CA-IDMS/VSAM Transparency that the exit modified the length of the record in the buffer.
134	EXIT-SET-ARGUMENT	A	1	Tells CA-IDMS/VSAM Transparency that the exit set the argument.
135	EXIT-SET-FEEDBACK	A	1	Tells CA-IDMS/VSAM Transparency that the exit set the feedback code. CA-IDMS/VSAM Transparency will display a message with this feedback code to the job log.
136	FILLER	A	2	Filler
138	FILLER	A	8	Filler
146	FILLER	A	2	Filler
148	EXP-SAVE-AREA	A	72	
218	FILLER	A	12	Filler

E.2.3 Sample COBOL user exit

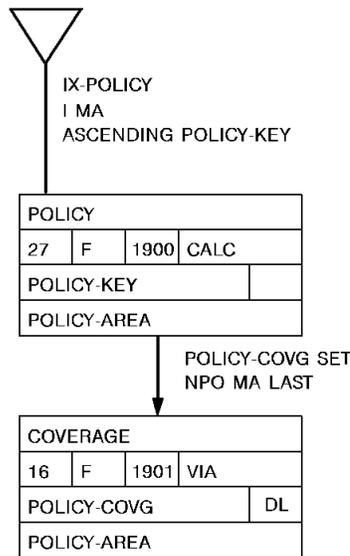
Functions: This is a sample CA-IDMS/VSAM Transparency user exit written in COBOL. It is functionally equivalent to the sample Assembler exit located later in this section. The source code for this program is provided on the installation tape with CA-IDMS/VSAM Transparency (member name ESVSXITC).

This exit allows you to convert the VSAM structure into the CA-IDMS/DB structure.

VSAM structure (variable length KSDS record)

Policy info.	Coverage info.	Coverage info.	Coverage info.	Coverage info.
--------------	----------------	----------------	----------------	----------------

CA-IDMS structure: The user exit must navigate the database and build the record that the VSAM application program is expecting.



Functions as a before exit: This exit is invoked both before and after any DML processing. As a **before exit**, it performs the following functions:

- Before a **PUT**, it moves zero to the control (OCCURS) field from a packed format to a binary format.
- Before a **GET**, it is not required.
- Before an **ERASE**, it obtains the policy record and does an ERASE ALL, erasing the POLICY and all COVERAGE records. The exit then sets a flag (EXB-RETURN-IMMED) to tell CA-IDMS/VSAM Transparency to bypass its processing and return immediately to the user.

Functions as an after exit: As an **after exit**, it performs the following functions:

- After a **PUT**, it stores all the dependent (MEMBER) records in the set.
- After a **GET**, it obtains all the dependent (MEMBER) records in the set and then converts the control (OCCURS) field from a binary format to a packed format.
- After an **ERASE**, it is not required.

Note: The POLICY record is the record defined in the FMT.

Sample COBOL exit program

```

IDENTIFICATION DIVISION.
PROGRAM-ID.  ESVSXITC.
ENVIRONMENT DIVISION.
  IDMS-CONTROL SECTION.
  PROTOCOL.   MODE IS BATCH  DEBUG
              IDMS-RECORDS MANUAL.
*
INPUT-OUTPUT SECTION.
*
DATA DIVISION.
*
SCHEMA SECTION.
DB  EXITSUB  WITHIN  EXITCHEM.
*
WORKING-STORAGE SECTION.
*
01  FILLER                                PIC X(8) VALUE 'WS START'.
*
COPY IDMS SUBSCHEMA-NAMES.
COPY IDMS SUBSCHEMA-RECORDS.
*
01  INDEX-1                              PIC 9(1) VALUE 0.
*

LINKAGE SECTION.
01  USER-EXIT-CONTROL-BLOCK.
    05  EXB-ID                            PIC X(4).
    05  EXB-USER                           PIC S9(8) COMP.
    05  EXB-ADDR-RWA                       PIC S9(8) COMP.
    05  EXB-ADDR-STACK                     PIC S9(8) COMP.
    05  EXB-ADDR-SSC                       PIC S9(8) COMP.
    05  FILLER                             PIC S9(8) COMP.
    05  EXB-ADDR-RPL                       PIC S9(8) COMP.
    05  EXB-ADDR-ARGUMENT                  PIC S9(8) COMP.
    05  EXB-ADDR-RECORD-BUFFER            PIC S9(8) COMP.
    05  EXB-MAX-KEY-LENGTH                 PIC S9(4) COMP.
    05  EXB-REC-BUFFER-LENGTH              PIC S9(4) COMP.
    05  EXB-VSAM-REC-LENGTH                PIC S9(4) COMP.
    05  EXB-VSAM-KEY-LENGTH                PIC S9(4) COMP.
    05  EXB-VSAM-KEY-POSITION              PIC S9(4) COMP.
    05  EXB-FMT-NAME                       PIC X(8).
    05  EXB-DD-NAME                        PIC X(8).
    05  EXB-IDMS-REC-NAME                  PIC X(16).
    05  EXB-IDMS-SET-NAME                  PIC X(16).
    05  EXB-RESERVED                       PIC S9(2) COMP.
    05  EXB-FEEDBACK                       PIC X(4).
    05  EXB-RPL-REQUEST-TYPE               PIC X(6).
        88  GET-REQUEST                     VALUE 'GET'.
        88  PUT-REQUEST                     VALUE 'PUT'.
        88  POINT-REQUEST                   VALUE 'POINT'.
        88  ENDREQ-REQUEST                  VALUE 'ENDREQ'.
        88  ERASE-REQUEST                   VALUE 'ERASE'.
    05  EXB-RPL-OPTION-1.
        10  EXB-DIRECT                      PIC X.

```

```
      88 DIRECT-REQUEST      VALUE 'X'.
10  EXB-SEQUENTIAL          PIC X.
      88 SEQUENTIAL-REQUEST VALUE 'X'.
10  EXB-SKIP                PIC X.
      88 SKIP-REQUEST        VALUE 'X'.
10  EXB-KEY-GT-EQ           PIC X.
      88 KEY-GT-EQ-REQUEST   VALUE 'X'.
10  EXB-GENERIC             PIC X.
      88 GENERIC-REQUEST     VALUE 'X'.
10  FILLER                  PIC X(3).
05  EXB-RPL-OPTION-2.
10  EXB-KEYED-ACCESS        PIC X.
      88 KEYED-ACCESS        VALUE 'X'.
10  EXB-ADDRESS-ACCESS     PIC X.
      88 ADDRESS-ACCESS      VALUE 'X'.
10  EXB-BACKWARD-ACCESS    PIC X.
      88 BACKWARD-ACCESS     VALUE 'X'.
10  EXB-LAST-REC-ACCESS    PIC X.
      88 LAST-REC-ACCESS     VALUE 'X'.
10  EXB-UPDATE-ACCESS      PIC X.
      88 UPDATE-ACCESS       VALUE 'X'.
10  EXB-SET-POSITION        PIC X.
      88 SET-POSITION        VALUE 'X'.
10  FILLER                  PIC X(2).
05  EXB-VSAM-FILE-TYPE.
10  EXB-KSDS                PIC X.
      88 KSDS                 VALUE 'X'.
10  EXB-PATH                PIC X.
      88 PATH                  VALUE 'X'.
10  EXB-RRDS                PIC X.
      88 RRDS                  VALUE 'X'.
10  EXB-ESDS                PIC X.
      88 ESDS                  VALUE 'X'.
10  EXB-BASE-CLUS-ESDS     PIC X.
      88 BASE-CLUS-ESDS      VALUE 'X'.
10  EXB-SET-DEFINED         PIC X.
      88 SET-DEFINED          VALUE 'X'.
10  EXB-NATIVE-VSAM        PIC X.
      88 NATIVE-VSAM          VALUE 'X'.
10  FILLER                  PIC X.
05  EXB-EXIT-FLAGS.
10  EXB-EXIT-TYPE          PIC X.
      88 BEFORE-EXIT          VALUE 'B'.
      88 AFTER-EXIT           VALUE 'A'.
10  EXB-SKIP-TO-AFTER      PIC X.
      88 SKIP-TO-AFTER        VALUE 'X'.
10  EXB-RETURN-IMMED       PIC X.
      88 RETURN-IMMED         VALUE 'X'.
10  EXB-EXIT-SET-REC-LENGTH PIC X.
      88 EXIT-SET-REC-LENGTH  VALUE 'X'.
10  EXB-EXIT-SET-ARGUMENT  PIC X.
```

```

            88 EXIT-SET-ARGUMENT      VALUE 'X'.
    10  EXB-EXIT-SET-FEEDBACK        PIC X.
            88 EXIT-SET-FEEDBACK     VALUE 'X'.
    10  FILLER                        PIC X(2).
    05  FILLER                        PIC X(8).
    05  FILLER                        PIC X(2).
    05  EXB-SAVE-AREA                PIC X(72).
    05  FILLER                        PIC X(12).
*
COPY IDMS SUBSCHEMA-CTRL.
*
01  IO-RECORD.
    05  POLICY-HEADER.
        10  POLICY-KEY1              PIC X(11).
        10  POLICY-TYPE              PIC 9(4).
    05  POL-DATA-1                  PIC X(10).
    05  POLICY-OCCURS                PIC S9(4) COMP.
    05  COV-DATA                    PIC X(10)
        OCCURS 4 TIMES.

PROCEDURE DIVISION
    USING EXIT-DSECT, SUBSCHEMA-CTRL, IO-RECORD.
BEGIN SECTION.
    IF BEFORE-EXIT AND PUT-REQUEST
        PERFORM BEFORE-PUT-EXIT
    ELSE
        IF BEFORE-EXIT AND ERASE-REQUEST
            PERFORM BEFORE-ERASE-EXIT
        ELSE
            IF AFTER-EXIT AND GET-REQUEST
                PERFORM AFTER-GET-EXIT
            ELSE
                IF AFTER-EXIT AND PUT-REQUEST
                    PERFORM AFTER-PUT-EXIT.

    GOBACK.
BEGIN-XIT.
EXIT.
BEFORE-PUT-EXIT SECTION.
    MOVE 0 TO POLICY-OCCURS.
BEFORE-PUT-EXIT-XIT.
EXIT.
BEFORE-ERASE-EXIT SECTION.
    BIND POLICY.
    IF NOT DB-STATUS-OK
        PERFORM SET-FEEDBACK.
    MOVE POLICY-HEADER TO POLICY-KEY.
    OBTAIN CALC POLICY.
    IF NOT DB-STATUS-OK
        PERFORM SET-FEEDBACK.

```

```
ERASE POLICY ALL.  
IF NOT DB-STATUS-OK  
    PERFORM SET-FEEDBACK.  
MOVE 'X' TO EXB-RETURN-IMMED.  
BEFORE-ERASE-EXIT-XIT.  
EXIT.  
AFTER-GET-EXIT SECTION.  
MOVE 0 TO INDEX-1.  
MOVE SPACES TO COV-DATA (1).  
MOVE SPACES TO COV-DATA (2).  
MOVE SPACES TO COV-DATA (3).  
MOVE SPACES TO COV-DATA (4).  
BIND COVERAGE.  
IF NOT DB-STATUS-OK  
    PERFORM SET-FEEDBACK.  
IF POLICY-TYPE = '1040'  
    PERFORM OBTAIN-COVERAGE-RECORDS 4 TIMES  
ELSE  
    IF POLICY-TYPE = '1041'  
        PERFORM OBTAIN-COVERAGE-RECORDS 3 TIMES.  
MOVE 'X' TO EXB-EXIT-SET-REC-LENGTH.  
AFTER-GET-EXIT-XIT.  
EXIT.  
OBTAIN-COVERAGE-RECORDS SECTION.  
ADD 1 TO INDEX-1.  
OBTAIN NEXT COVERAGE WITHIN POLICY-COVG.  
IF DB-STATUS-OK  
    MOVE COVERAGE TO COV-DATA (INDEX-1)  
ELSE  
    PERFORM SET-FEEDBACK.  
OBTAIN-COVERAGE-RECORDS-EXIT.  
EXIT.  
AFTER-PUT-EXIT SECTION.  
MOVE 0 TO INDEX-1.  
BIND COVERAGE.  
IF NOT DB-STATUS-OK  
    PERFORM SET-FEEDBACK.  
IF POLICY-TYPE = '1040'  
    PERFORM PUT-COVERAGE-RECORDS 4 TIMES  
ELSE  
    IF POLICY-TYPE = '1041'  
        PERFORM PUT-COVERAGE-RECORDS 3 TIMES.  
AFTER-PUT-EXIT-XIT.  
EXIT.  
PUT-COVERAGE-RECORDS SECTION.  
ADD 1 TO INDEX-1.  
MOVE COV-DATA (INDEX-1) TO COVERAGE.  
STORE COVERAGE.  
IF NOT DB-STATUS-OK  
    PERFORM SET-FEEDBACK.  
PUT-COVERAGE-RECORDS-EXIT.  
EXIT.  
SET-FEEDBACK SECTION.  
MOVE ERROR-STATUS TO EXB-FEEDBACK.  
MOVE 'X' TO EXB-EXIT-SET-FEEDBACK.  
GOBACK.  
SET-FEEDBACK-XIT.  
EXIT.
```

E.3 Assembler user exit program

When programming an Assembler user exit for CA-IDMS/VSAM Transparency, fulfill the requirements noted below. This discussion is followed by the control block layout (DSECT), a user exit template, a sample Assembler user exit, and a discussion of the CA-IDMS/DB-supplied macros available for use with Assembler user exits.

E.3.1 Program requirements

Parameter list address: A parameter list address in register 1 points to the following addresses:

- The exit control block
- The subschema control block
- The record buffer

Reentrant exit: The exit must be reentrant. You may elect to issue a #GETSTG to obtain a work area for the program to use and a #FREESTG to release the storage prior to the end of the exit. See E.3.5, “Macros” on page E-29 for more information.

@MODE compiler directive statement: In the @MODE compiler directive statement, specify MODE=BATCH even if the exit is to run under the DC/UCF environment.

Subschema control DSECT: Copy the subschema control DSECT (#SSCDS). In addition to this DSECT, an additional field is required for expansion of DML macros. The field definition is:

```
DMLSEQ DC F'0' DML SEQUENCE NUMBER FOR DEBUGGING
```

Issue a @BIND: CA-IDMS/VSAM Transparency binds the record defined in the FMT the first time it attempts to access it. The exit must issue a bind only for additional records it needs. Issue a @BIND statement for the first access of each additional record:

```
@BIND REC=recname,IOAREA=rec-location
```

Issue CA-IDMS/DB verbs: After the bind is complete you can issue CA-IDMS/DB verbs to access the record. For a complete description of all DML verbs see *CA-IDMS DML Reference - Assembler*.

Assemble and link edit requirements

```
INCLUDE libname(ESVSIDMS)
INCLUDE libname(IDMSBALI)
```

Note: Libname is the DD name of the file in the JCL that contains the CA-IDMS/DB-supplied object modules.

Defining the exit to CA-IDMS/DB: Include an ADD PROGRAM statement for the exit program in the system generation. The ADD PROGRAM statement should appear as follows:

```
ADD PROGRAM exitname
  LANGUAGE IS ASSEMBLER
  REENTRANT
  NOPROTECT.
```

Note: The exit accesses DC/UCF system control blocks, therefore storage protection must be off (NOPROTECT).

E.3.2 Assembler user exit control block

Field descriptions: In the Type column, C means character, F means fullword binary, and H means halfword binary. The macro is provided in the CA-IDMS/DB macro library.

Hex Offset	Name	Type	Length	Description
0	EXBID	C	4	Contains the literal 'EXB'
4	EXBUSER	F	4	Can be used by the exit
8	EXBARWA	F	4	Exit must not modify this field
C	EXBSTACK	F	4	
10	EXBSSCA	F	4	Address of the Subschema Control
14	FILLER	F	4	Filler
18	EXBRPLA	F	4	
1C	EXBARGA	F	4	
20	EXBBUFA	F	4	Address of the record in the buffer
24	EXBMXKL	H	2	Maximum key length.
26	EXBBUFL	H	2	Current length of record in the record buffer
28	EXBRECL	H	2	Contains the VSAM record length obtained from FMT
2A	EXBKEYL	H	2	Contains the VSAM key length obtained from FMT
2C	EXBKEYP	H	2	Contains the VSAM key position obtained from FMT
2E	EXBFMTNM	C	8	FMT name
36	EXBDDNM	C	8	DD name
3E	EXBRECNM	C	16	CA-IDMS/DB record name

Hex Offset	Name	Type	Length	Description
4E	EXBSETNM	C	16	CA-IDMS/DB set name
5E	EXBRESV	H	2	Reserved field
60	EXBFDBK	C	4	Allows you to set a feedback code (return code) from the exit program. By placing a value here (such as subschema control status) and by moving an 'X' to EXIT-SET-FEEDBACK, CA-IDMS/VSAM Transparency will display an error message on the job log including the feedback code you set.
60	EXBERID	C	1	Internal error id
61	EXBRTCD	C	1	Return code
62	EXBCOMP	C	1	Component code
63	EXBERCD	C	1	Error code
64	EXBRTYPE	C	6	Identifies the type of VSAM call issued. Use this field in conjunction with EXB-EXIT-FLAGS to determine what type of processing will be required. Valid values are: <ul style="list-style-type: none"> ■ 'GET ' ■ 'PUT ' ■ 'POINT ' ■ 'ENDREQ' ■ 'ERASE '
6A	EXBOPT1	C	8	Further qualifies the request type:
6A	EXBDIR	C	1	Direct request
6B	EXBSEQ	C	1	Sequential request
6C	EXBSKP	C	1	Skip request
6D	EXBKGE	C	1	Key gt/eq request
6E	EXBGEN	C	1	Generic request
6F	FILLER	C	3	Filler
72	EXBOPT2	C	8	Further qualifies the request type:
72	EXBKEYA	C	1	Keyed access

Hex Offset	Name	Type	Length	Description
73	EXBADDA	C	1	Address access
74	EXBBWD	C	1	Backward access
75	EXBLRD	C	1	Last record access
76	EXBUPD	C	1	Update access
77	EXBNSP	C	1	Set position
78	FILLER	C	2	Filler
7A	EXBFTYPE	C	8	Identifies the VSAM file type:
7A	EXBKSDS	C	1	KSDS
7B	EXBPATH	C	1	PATH
7C	EXBRRDS	C	1	RRDS
7D	EXBESDS	C	1	ESDS
7E	EXBBCES	C	1	Base cluster is ESDS
7F	EXBSET	C	1	Set is defined
80	EXBNVSAM	C	1	Data is in native VSAM
81	FILLER	C	1	Filler
82	EXBXFLAG	C	8	Exit flags that describe exit processing:
82	EXBEXIT	C	1	The first byte identifies the exit type: <ul style="list-style-type: none"> ■ Before = B ■ After = A
The next 5 bytes describe exit processing. Put an X in the byte to turn the character switch on. The default is blank.				
83	EXBSKPA	C	1	Skip to after tells CA-IDMS/VSAM Transparency to invoke the after exit immediately after the before exit and bypass its processing.
84	EXBXRETN	C	1	Return immediately tells CA-IDMS/VSAM Transparency to return immediately to the user and bypass its processing.

Hex Offset	Name	Type	Length	Description
85	EXBSETLN	C	1	Exit set record length tells CA-IDMS/VSAM Transparency the exit modified the length of the record in the buffer.
86	EXBSETAR	C	1	Exit set argument tells CA-IDMS/VSAM Transparency the exit set argument.
87	EXBSETFB	C	1	Exit set feedback tells CA-IDMS/VSAM Transparency the exit set the feedback code. CA-IDMS/VSAM Transparency will display a message with this feedback code to the job log.
88	FILLER	C	2	Filler
8A	FILLER	C	8	Filler
94	EXBSAVE	C	72	Exit save area
DC	FILLER	C	12	Filler

E.3.3 Assembler user exit template

Assembler exits for CA-IDMS/VSAM Transparency use standard OS/390 and VSE/ESA linkage. Use the following code as a template. Insert your functional code as indicated.

```

EXITNAME CSECT
#MOPT ENV=USER          SET EXIT TO USER MODE
@MODE MODE=BATCH,QUOTES=YES,DEBUG=YES
*
STM R14,R12,12(R13)    SAVE CALLER'S REGISTERS
BALR R12,0             ESTABLISH R12 AS BASE
USING *,R12
B START               BRANCH AROUND LITERAL
DC CL8'EXITNAME'      EYECATCHER
USING SSC,R2          SUBSCHEMA CONTROL
USING EXBDS,R3        EXIT DSECT ADDRESSABILITY
USING RECORD,R8       USER RECORD ADDRESSABILITY
USING CSA,R10         COMMON SYSTEM AREA
START LR R7,R13        SAVE R13
L R3,0(R1)            R3 → EXB
L R2,4(R1)            R2 → SSC
L R8,8(R1)            R8 → RECORD BUFFER
LA R13,EXBSAVE        R13 → USER SAVE AREA
ST R13,4(0,R13)       BACKCHAIN SAVE AREA
*
*
insert your functional code here
*
*
RTN EQU *
L R13,4(0,R13)        RESTORE ADDRESS OF R13 SAVE AREA
LM R14,R12,12(R13)   RESTORE CALLER'S REGISTERS
SR R15,R15            ZERO RETURN CODE
BR R14                RETURN TO CALLER

```

E.3.4 Sample assembler user exit

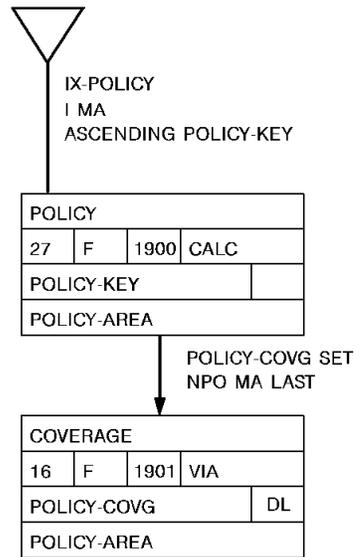
Functions: This is a sample CA-IDMS/VSAM Transparency user exit written in Assembler. It is functionally equivalent to the sample COBOL exit in E.2.3, “Sample COBOL user exit” on page E-11. The source code for this program is provided on the installation tape with CA-IDMS/VSAM Transparency (member name ESVSXITA).

This exit allows you to convert the VSAM structure into CA-IDMS/DB structure.

VSAM structure (variable length KSDS record)

Policy info.	Coverage info.	Coverage info.	Coverage info.	Coverage info.
--------------	----------------	----------------	----------------	----------------

CA-IDMS structure: The user exit must navigate the database and build the record that the VSAM application program is expecting.



Functions as before exit: This exit is invoked both before and after any DML processing. As a **before exit**, it performs the following functions:

- Before a **PUT**, it moves zeroes to the control (OCCURS) field.
- Before a **GET**, it is not required.
- Before an **ERASE**, it obtains the **POLICY** record and obtains and erases all **COVERAGE** records.

Functions as after exit: As an **after exit**, it performs the following functions:

- After a **PUT**, it stores all the dependent (MEMBER) records in the set.
- After a **GET**, it obtains all the dependent (MEMBER) records in the set and then converts the control (OCCURS) field from a binary format to a packed format.
- After an **ERASE**, it is not required.

Sample Assembler exit

E.3 Assembler user exit program

```

***
*** THIS EXIT IS INVOKED USING STANDARD OS/390 LINKAGE; *
*** BALR 14,15 (FROM CALLING PROGRAM) *
*** STM 14,12,12(13) (FIRST THING IN CALLED PROGRAM) *
*** BR 14 (TO RETURN) *
***
*** REGISTER USAGE CONVENTIONS ARE: *
***
*** R1 --> SET UP TO SSC FOR EXITIDMS *
*** R2 --> SUBSCHEMA CONTROL DSECT *
*** R3 --> EXIT DSECT *
*** R4 --> COVERAGE RECORD DSECT/WORK REGISTER *
*** R5 --> WORK AREA FOR MOVE OF RECORD *
*** R6 --> WORK REGISTER *
*** R7 --> WORK AREA FOR CONVERSION *
*** R8 --> POLICY RECORD DSECT *
*** R9 --> TCE *
*** R10 --> CSA *
*** R11 --> SAVE AREA *
*** R12 --> BASE REGISTER *
*** R13 --> SAVE AREA *
*** R14 --> USED BY CALL CONVENTION *
*** R15 --> USED BY CALL CONVENTIONS *
***
*** INPUT: *
***
*** R1 --> PARAMETER LIST *
*** R1+0 --> ADDRESS OF EXB *
*** R1+4 --> ADDRESS OF SSC *
*** R1+8 --> ADDRESS OF RECORD BUFFER *
*****
EJECT
ESVSXITA CSECT
#MOPT ENV=USER
@MODE MODE=BATCH,QUOTES=YES,DEBUG=YES
* SPECIFY OPERATING ENVIRONMENT,
* NOTATION CONVENTION, DEBUG OPTION
BEGIN STM R14,R12,12(R13) SAVE CALLER'S REGISTERS
BALR R12,0 ESTABLISH R12 AS BASE
USING *,R12
SPACE
B START BRANCH AROUND LITERAL
*
DC CL8'ESVSXITA' EYECATCHER
*
USING SSC,R2 SUBSCHEMA CONTROL ADDRESSIBILITY
USING EXBDS,R3 EXIT DSECT ADDRESSABILITY
USING COVERAGE,R4 COVERAGE RECORD ADDRESSIBILITY
USING RECORD,R8 POLICY RECORD ADDRESSIBILITY
USING CSA,R10 COMMON SYSTEM AREA ADDRESSIBILITY
SPACE
START EQU *
LR R7,R13 SAVE R13
L R3,0(R1) R3 -> EXB
L R2,4(R1) R2 -> SSC
L R8,8(R1) R8 -> RECORD BUFFER
LA R13,EXBSAVE R13 -> USER SAVE AREA
SPACE

```

```

***
*** ACQUIRE STORAGE FOR EXIT TO USE
***
GETSTG #GETSTG TYPE=(USER,SHORT),PLIST=*,LEN=50,INIT=0,          X
        ADDR=(R4),ERROR=STGERROR
        SPACE
*
        EJECT
* CHECK TO SEE IF THIS IS A BEFORE EXIT WITH A PUT REQUEST
CHKBEFOR CLI  EXBXEXIT,C'B'          CALLING BEFORE EXIT??
        BNE  CHKAFTER                NO, CHECK FOR AFTER EXIT
        CLC  EXBRTYPE,=CL6'PUT'      IS THIS A PUT REQUEST??
        BNE  CHKERASE                NO, DON'T DO ANYTHING ELSE
        SPACE
* BEFORE EXIT - PUT REQUEST
* MOVE ZERO TO CONTROL FIELD
BEFORPUT MVC  CNTFLD,=H'00'          MOVE ZERO TO OCCURS FIELD
        B    RTN                    RETURN
        EJECT
* CHECK TO SEE IF THIS IS AN BEFORE EXIT WITH A ERASE REQUEST
CHKERASE CLI  EXBXEXIT,C'B'          YES, CALLING BEFORE EXIT??
        BNE  RTN                    NO, DON'T DO ANYTHING ELSE
        CLC  EXBRTYPE,=CL6'ERASE'    IS THIS AN ERASE REQUEST??
        BNE  RTN                    NO, DON'T DO ANYTHING ELSE
        SPACE
* BEFORE EXIT - ERASE REQUEST
* GET POLICY RECORD AND ERASE ALL COVERAGE RECORDS WITHIN
* POLICY-COVG SET.
        SPACE
BEFORERA EQU  *
        CLC  POLTYP,=CL4'1040'        IS RECORD TYPE 1040??
        BNE  REC1041                NO, CHECK FOR TYPE 1041
        LA  R6,4                    R6 → 4 COVERAGE RECORDS TO GET
        B    BINDPOL                GO BIND POLICY RECORD
REC1041 CLC  POLTYP,=CL4'1041'        IS RECORD TYPE 1041??
        BNE  RTN                    NO, DON'T DO ANYTHING ELSE
        LA  R6,3                    R6 → 3 COVERAGE RECORDS TO GET
        SPACE
*** BIND POLICY RECORD
BINDPOL EQU  0
        @BIND REC=POLREC,IOAREA=(R4)  BIND COVERAGE RECORD TO R4
        CLC  SSCSTAT,=CL4'0000'      STATUS ZERO
        BNE  RTNERR                NO, RETURN ERROR
        SPACE
*** OBTAIN POLICY RECORD
OBTPOLCY @FIND  CALC,REC=POLREC      OBTAIN CALC POLICY RECORD
*
        CLC  SSCSTAT,=CL4'0000'      STATUS ZERO??
        BNE  RTNERR                NO, RETURN ERROR
*** BIND COVERAGE RECORD
        @BIND REC=COVGREC,IOAREA=(R4)  BIND COVERAGE RECORD TO R4
        CLC  SSCSTAT,=CL4'0000'      STATUS ZERO

```

```

        BNE  RTNERR          NO, RETURN ERROR
        SPACE
***  OBTAIN COVERAGE RECORDS
OBTCOVG @OBTAIN NEXT,SET=SETNAME,REC=COVGREC
*
*          OBTAIN ALL COVERAGE RECORDS FOR
          THIS POLICY
        CLC  SSCSTAT,=CL4'0000'  STATUS ZERO??
        BNE  RTNERR          NO, RETURN ERROR
        SPACE
***  ERASE COVERAGE RECORD
@ERASE ALL,REC=COVGREC  ERASE COVERAGE RECORD FOR
*
*          THIS POLICY
        CLC  SSCSTAT,=CL4'0000'  STATUS ZERO??
        BNE  RTNERR          NO, RETURN ERROR
        BCT  R6,OBTCOVG      WALK THE SET, OBTAIN THE NEXT
        SPACE
        B    RTN            RETURN
        EJECT
*  CHECK TO SEE IF THIS IS AN AFTER EXIT WITH A GET REQUEST
CHKAFTER CLI  EXBXEXIT,C'A'    YES, CALLING AFTER EXIT??
        BNE  RTN            NO, DON'T DO ANYTHING ELSE
        CLC  EXBRTYPE,=CL6'GET' IS THIS A GET REQUEST??
        BNE  CHKPUT        NO, CHECK FOR PUT REQUEST
        CLC  SSCSTAT,=CL4'0000'  STATUS OF LAST CALL = ZERO??
        BNE  RTN            NO, RETURN
        SPACE
*  AFTER EXIT - GET REQUEST
*  GET MEMBERS IN IDMS SET AND BUILD VSAM RECORD IN BUFFER
*  WE HAVE THE OWNER RECORD. OBTAIN ALL MEMBERS WITHIN
*  POLICY-COVG SET, AND BUILD VSAM TYPE RECORD FOR PROGRAM.
AFTERGET EQU  *
        CLC  POLTYP,=CL4'1040'  IS RECORD TYPE 1040??
        BNE  TYP1041        NO, CHECK FOR TYPE 1041
        LA   R6,4           R6 → 4 COVERAGE RECORDS TO GET
        B    SETUP        GO OBTAIN THE RECORDS
TYP1041 CLC  POLTYP,=CL4'1041'  IS RECORD TYPE 1041??
        BNE  NOTREC        NO, DON'T DO ANYTHING ELSE
        LA   R6,3           R6 → 3 COVERAGE RECORDS TO GET
        SPACE
SETUP    MVC  OCCURS,=CL40' '  CLEAR OUT AREA TO STORE COVERAGE
        LA   R5,27(0,R8)    R5 → START OF COVERAGE RECORDS
*
*          IN POLICY RECORD
***  BIND COVERAGE RECORD
@BIND REC=COVGREC,IOAREA=(R4)  BIND COVERAGE RECORD TO R4
        CLC  SSCSTAT,=CL4'0000'  STATUS ZERO
        BNE  RTNERR          NO, RETURN ERROR
        SPACE

```

```

*** OBTAIN COVERAGE RECORD
GETREC @OBTAIN NEXT,SET=SETNAME      OBTAIN NEXT COVERAGE RECORD
*                                     WITHIN POLICY-COVG SET
      CLC   SSCSTAT,=CL4'0000'      STATUS ZERO??
      BNE   RTNERR                   NO, RETURN ERROR
      MVC   0(10,R5),COVERAGE      MOVE COVERAGE REC INTO POLICY REC
      LA    R5,10(0,R5)             R5 → LOC OF NEXT COVERAGE RECORD
      BCT   R6,GETREC               WALK THE SET, OBTAIN THE NEXT
*   MOVE ZERO TO CONTROL FIELD
PACK   MVC   CNTFLD,=PL2'00'        MOVE ZERO TO OCCURS FIELD
      MVI   EXBSETLN,C'X'           HAVE PRPL SET NEW RECORD LENGTH
      B     RTN                     RETURN
      EJECT
*   CHECK TO SEE IF THIS A PUT REQUEST AND CALLING AFTER EXIT
CHKPUT CLI   EXBXEXIT,C'A'          CALLING AFTER EXIT??
      BNE   RTN                     NO, DON'T DO ANYTHING ELSE
      CLC   EXBRTYPE,=CL6'PUT'      IS THIS A PUT REQUEST??
      BNE   RTN                     NO, DON'T DO ANYTHING ELSE
      CLC   SSCSTAT,=CL4'0000'      STATUS OF LAST CALL = ZERO??
      BNE   RTN                     NO
      SPACE
*   AFTER EXIT - PUT REQUEST
*   STORE COVERAGE (MEMBER) RECORDS WITHIN POLICY-COVG SET
*   WE HAVE THE OWNER RECORD.  STORE ALL MEMBERS WITHIN POLICY-COVG
*   SET.
AFTERPUT EQU *
      CLC   POLTYP,=CL4'1040'       IS RECORD TYPE 1040??
      BNE   TYPE1041               NO, RECORD CHECK FOR TYPE 1041
      LA    R6,4                    R6 → 4 COVERAGE RECORDS TO STORE
      B     SETUPS                  GO STORE THE RECORDS
TYPE1041 CLC   POLTYP,=CL4'1041'    IS RECORD TYPE 1041??
      BNE   RTN                     NO, DON'T DO ANYTHING ELSE
      LA    R6,3                    R6 → 3 COVERAGE RECORDS TO STORE
      SPACE
SETUPS  LA    R5,27(0,R8)           R5 → START OF COVERAGE RECORDS
*                                     IN POLICY RECORD
*** BIND COVERAGE RECORD
      @BIND REC=COVGREC,IOAREA=(R4) BIND COVERAGE RECORD TO R4
      CLC   SSCSTAT,=CL4'0000'      STATUS ZERO
      BNE   RTNERR                   NO, RETURN ERROR
      SPACE
*** STORE COVERAGE RECORD
PUTREC MVC   DATA,0(R5)           MOVE POLICY/COVG INFO TO COVG REC
      @STORE REC=COVGREC           STORE COVERAGE RECORD
      SPACE
      CLC   SSCSTAT,=CL4'0000'      STATUS ZERO
      BNE   RTNERR                   NO, RETURN ERROR
      LA    R5,10(0,R5)             R5 → LOC OF NEXT COVERAGE RECORD
      BCT   R6,PUTREC               STORE THE NEXT RECORD
      B     RTN                     RETURN
      EJECT

```

```

NOTREC  MVC   OCCURS,=CL40'  '    CLEAR OUT AREA TO STORE COVERAGE
        SPACE
RTNERR  EQU   *
        MVI   EXBSETFB,C'X'      TELL ESVS EXIT SET FEEDBACK CODE
        MVC   EXBFDBK,SSCSTAT    MOVE SSCSTAT TO EXBFDBK
RTN     EQU   *
        #FREESTG ADDR=(R4)       FREE ACQUIRED STORAGE
        LR    R13,R7             RESTORE R13
        LM    R14,R12,12(R13)    RESTORE CALLER'S REGISTERS
        SR    R15,R15           ZERO RETURN CODE
        BR    R14               RETURN TO CALLER
        SPACE
STGERROR EQU  *
        DC    H'0'              ABEND WITH 0C1
        BR    R14               RETURN TO CALLER
        SPACE
        EJECT
STARS   DC    CL8'*****'      END OF EXECUTABLE CODE

DMLSEQ  DC    F'0'              NEEDED AT DML EXPANSION TIME
SETNAME DC    CL16'POLICY-COVG'  SET-NAME
COVGREC DC    CL16'COVERAGE'    RECORD-NAME
POLREC  DC    CL16'POLICY'      RECORD-NAME
        LTOrg
RECORD  DSECT
        DS    0F
POLICY  DS    0CL27
POLKEY  DS    0CL15
POLKEYDA DS CL11
POLTYP  DS    CL4
POLDAT  DS    CL10
CNTLFLD DS    CL2
OCCURS  DS    0CL40
COVG1   DS    CL10
COVG2   DS    CL10
COVG3   DS    CL10
COVG4   DS    CL10
        SPACE
COVERAGE DSECT
DATA    DS    CL10
        EJECT
        COPY  #EVXBDS
        EJECT
        COPY  #SSCDS
        EJECT
        PRINT OFF
        PRINT NOGEN
        COPY  #CSADS
        PRINT GEN
        END   ESVSXITA

```

E.3.5 Macros

There are three macros supplied with CA-IDMS/DB that can be used in Assembler:

#GETSTG

- #GETSTG acquires variable storage from a storage pool or obtains the address of a previously acquired storage area. Syntax is shown below:

```
{label} #GETSTG TYPE=(USER,SHORT),PLIST=*,
              LEN=stg-len,INIT=init-val,
              ADDR=(R1),ERROR=error-label
```

- TYPE is used to indicate the type of storage acquired.
- PLIST indicates where the 6-fullword #GETSTG parameter list will be built. * indicates that the list will be built inline.
- LEN specifies the size, in bytes, of a new storage area. *Stg-len* is an absolute expression.
- INIT specifies an initial value for the storage. *Init-val* is an absolute expression of the initial value.
- ADDR=(R1) specifies where CA-IDMS/DC will return the address of the acquired storage.
- ERROR specifies the symbolic name of the routine to which control should be returned to in case of an error.

Note: If the user exit is acquiring storage, a #FREESTG must be done before the end of the program.

#FREESTG:

- #FREESTG requests CA-IDMS/DC to release a block of variable storage. Syntax is shown below:

```
{label} #FREESTG ADDR=(R1)
```

- ADDR=(R1) specifies the address of the storage area to be released.

#MOPT:

- #MOPT sets up the options for the issuing module. The only code that should appear prior to the #MOPT is 'TITLE', COMMENTS, or source macro definitions. Syntax is shown below:

```
#MOPT ENV=USER
```

- ENV=USER specifies that this is a DC/UCF user module.

Note: The #MOPT macro also equates register 0 through register 15 to R0 through R15.

For a complete definition of these macros see *CA-IDMS DML Reference - Assembler*.

Appendix F. TSO File Allocation with CA-IDMS/VSAM Transparency

F.1 About this appendix	F-3
F.2 TSO file allocation syntax	F-4

F.1 About this appendix

This appendix provides the syntax and parameter descriptions used under TSO file allocation.

F.2 TSO file allocation syntax

Purpose Under TSO, each CA-IDMS/VSAM Transparency file must be directed to the subsystem. The ESVSALLO command allocates a CA-IDMS/VSAM Transparency file under TSO. It establishes a connection to the CA-IDMS/VSAM Transparency subsystem and defines the attributes of the CA-IDMS/VSAM Transparency file.

Syntax

```

>> ESVSALLO ( FILE(filename) DDNAME(filename) )
> FMT (fmtname)
> SSc (subschema-name)
> SSubsys ( ( ESVS ← subsystem-name ) )
> Rbufsz ( ( 512 ← nnnnn ) )
> DBname (database-name)
> Nodename (nodename)
> DICTNAme (dictionary-name)
> DICTNOde (nodename)
> MSg ( ( LOG ← CONS OFF ) )
> STats ( ( FILE ON ) )
> Trace ( ( nnnn WRAP LOG ) )

```

Parameters

FILE(filename)/DDNAME(filename)

Specifies the CA-IDMS/VSAM Transparency file. *Filename* is a 1- to 8-character name that must be included in each invocation of the command.

FMT(fmtname)

Specifies the name of the file management table that defines the file to be escaped. *Fmtname* is a 1- to 8-character name that must be included in each invocation of the command.

SSc(subschema-name)

Specifies the name of the subschema to be used with the CA-IDMS/VSAM Transparency program. You must specify the subschema for the first file allocated; it is optional for subsequent files. The subschema specified for the first file allocated is used for all subsequent file allocations (whether or not a different subschema is specified) unless all CA-IDMS/VSAM Transparency file allocations are freed within TSO. Then, the last subschema defined on an allocation will be the one used.

SUBsys(subsystem-name)

Specifies the name of the CA-IDMS/VSAM Transparency subsystem. The default is ESVS.

Rbufsz(nnnnn)

Specifies the size of the buffer that CA-IDMS/VSAM Transparency uses for communication between the front end and the back end. The default is 512. *Nnnnn* is a 1- to 5-digit number that is valid for central version only. RBUFSZ must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

DBname(database-name)

Specifies the name of the CA-IDMS/DB database. DBNAME must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

Nodename(nodename)

Specifies the nodename of a DC/UCF system. NODENAME must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

DICTName(dictionary-name)

Specifies the name of the data dictionary containing the subschema. DICTNAME must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

DICTNode(nodename)

Specifies the location of the dictionary. DICTNODE must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

MSg(**LOG/CONS/OFF**)

Specifies the destination of the CA-IDMS/VSAM Transparency key feedback messages. LOG is the default.

CONS writes the feedback messages to the console and the log.

OFF turns off the display of feedback messages.

MSG must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

STats(FILE/ON)

Requests that database statistics be collected for all CA-IDMS/VSAM Transparency files accessed. ON places statistics for all files together on the log file. FILE places statistics for each file separately on the log file. STATS must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

Trace(nnnn,WRAP/LOG)

Requests that a trace of calls made by the CA-IDMS/VSAM Transparency program be displayed on the system log.

Nnnn is the number of fullwords allowed in the trace table and can be a 1- to 4-digit number.

WRAP allows the trace table to wrap around when the limit is reached.

LOG writes the trace table to the log and clears the trace table when the limit is reached. TRACE must be specified before the first file is opened (readied). It may be re-specified on subsequent file allocation statements as long as all previously allocated files have been closed.

For additional information on these parameters see Chapter 5, “Runtime Operations” on page 5-1.

Example: The following example allocates two files and their file management tables. Both files use the same subschema.

```
ESVSALLO FILE(TESTFILE) FMT(TESTFMT) SSC(TESTSSC)
ESVSALLO DDNAME(TESTFIL2) FMT(TESTFMT2) MSG(CONS)
```

Support: TSO file allocation support consists of two modules:

- Command processor (ESVSALLO)

The TSO file allocation command (ESVSALLO) must be linked into SYS1.CMDLIB or any other library concatenated to the SYS1.CMDLIB data set.

- Help data set (ESVSALLH)

The help dataset (ESVSALLH) must be renamed to ESVSALLO and moved into SYS1.HELP or any other library concatenated to the SYS1.HELP data set using the IEBUPDTE utility.

▶▶

Appendix G. VSE/ESA CICS SYSESVS Parameters

G.1 About this appendix G-3
G.2 SYSESVS parameter file G-4

G.1 About this appendix

This appendix provides information about VSE/ESA CICS SYSESVS parameters.

G.2 SYSESVS parameter file

Purpose: At VSE/ESA CICS VSAM/T startup, CA-IDMS reads a SYSESVS parameter file. The purpose of this file is to accommodate all the VSAM/T parameters required to run VSAM/T transactions in a VSE/ESA CICS environment.

What the file contains: The SYSESVS parameter file consists of 80-character records which can contain any number of keyword operands and keyword values. The file contains the following types of parameters:

- **Global parameters** — Describe VSAM/T characteristics that will be in affect for all VSAM/T activity
- **File-specific parameters** — Specify the file name and file management table for each VSAM/T file defined to the CICS system.

The table below provides the layout of the SYSESVS record:

Cols	Description
1-72	Keyword operands and keyword values
73-80	Reserved for record sequence numbers

What to consider: If a '--' character string is encountered in any position in a SYSESVS line, the rest of the line is treated as comments. Any number of blanks, commas, or semicolons can separate the keyword operands.

Global parameters: The SYSESVS keyword operands and their values are:

CWADISP=nnnn

Specifies the 1- to 4-digit value equal to the CWADISP specified in the assembly of IDMSINTC. The CWADISP parameter is required and its value is typically around 16.

DBNAME=xxxxxxxx

Specifies the 1- to 8-character name of the default database to be used with all CICS VSAM/T transactions. This parameter is used in a multiple-database environment. You can override this parameter for each CICS VSAM/T transaction by using the transaction name table (TNT).

NODENAME=xxxxxxxx

Specifies the 1- to 8-character default DDS node to be used with all CICS VSAM/T transactions. This parameter is used in a multiple-database environment. You can override this parameter for each CICS VSAM/T transaction by using the transaction name table (TNT).

DICTNAME=xxxxxxxx

Specifies the 1- to 8-character name of the data dictionary containing the VSAM/T SUBSCHEMA. This parameter is used in a multiple-dictionary environment.

You can override this parameter for each CICS VSAM/T transaction by using the transaction name table (TNT).

DICTNODE=xxxxxxxx

Specifies the 1- to 8-character location of the data dictionary containing the VSAM/T SUBSCHEMA. This parameter is used in a multiple-dictionary environment. You can override this parameter for each CICS VSAM/T transaction by using the transaction name table (TNT).

SUBSCHEMA=xxxxxxxx

Specifies the 1- to 8-character default subschema name to be used with all CICS VSAM/T transactions. This parameter is required. You can override this parameter for each CICS VSAM/T transaction by using the transaction name table (TNT).

TNT=xxxxxxxx

Specifies the 1- to 8-character transaction name table that will be used when processing CICS VSAM/T transactions. If all transactions will use the same subschema, this parameter is optional. If you use more than one subschema, this parameter is required.

RBUFSZ=nnnnn

Specifies the 1- to 5-digit size of the buffer that VSAM/T uses for communication between the front end and the back end. The default buffer size is 1024 bytes.

MSG=LOG/CONS/ALL/OFF

Displays the VSAM/T key feedback (run-time) messages during program execution. Values are:

LOG

Writes the key feedback messages to the log file (default)

CONS

Writes the key feedback messages to the operator console

ALL

Writes all RPL messages to the log file

OFF

Turns off the display of key feedback messages and RPL messages

VSAM/T writes key feedback messages for any open call, close call, or call that is in error, in the the following format:

FDBK=nnnnnnnn

Nnnnnnnn is an 8-byte field that contains the key feedback message code.

STATS=ON/FILE

Requests that database statistics be collected for all CICS VSAM/T transactions. Values are:

ON

Places statistics for all VSAM/T files together on the system log

FILE

Places the statistics for each VSAM/T file separately on the system log

OCMSGS=ALL/ERROR

Specifies which messages you want to appear on the console. Values are:

ERROR

Indicates that only error messages will appear on the console (default)

ALL

Indicates that both informational and error messages will appear on the console

TRACE=nnnn

Requests that a trace of calls made by the VSAM/T application program be displayed on the system log, where *nnnn* is the 1- to 4-digit number of fullwords allowed in the trace table.

TRACETYPE=WRAP/LOG

Specifies the trace options to be in affect when the TRACE parameter has been specified. Values are:

WRAP

Enables the trace table to wrap around if the fullword limit is reached (default). This means that subsequent program calls will be recorded over the existing trace table.

LOG

Writes the trace table to CDMSLOG and then clears the trace table, if the fullword limit for the trace table is reached. This means that subsequent program calls will be recorded on the cleared trace table.

File-specific parameters: You must specify the following parameters for each VSAM/T file that is to be processed during the CICS session:

FILENAME=xxxxxxx

Specifies a 1- to 7-character FILENAME of a VSAM/T file that is to be processed during this CICS session.

FMT=xxxxxxx

Specifies the 1 to 8-character file management table that defines the VSAM/T file characteristics for the file name specified in the FILENAME parameter, where *xxxxxxx* is the table name. This parameter is required and must follow the associated FILENAME parameter immediately.

Appendix H. IDMSLBLS Procedure

- H.1 About this appendix H-3
 - H.1.1 IDMSLBLS Procedure H-3

H.1 About this appendix

This appendix describes the IDMSLBLS Procedure which is provided on the VSE/ESA installation tape.

H.1.1 IDMSLBLS Procedure

What is the IDMSLBLS procedure?: IDMSLBLS is a procedure provided during a CA-IDMS VSE/ESA installation. It contains file definitions for the CA-IDMS components listed below. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
- SYSIDMS file

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in VSE/ESA JCL job streams.

The sample VSE/ESA JCL provided in this document includes the IDMSLBLS procedure. Therefore, individual file definitions for CA-IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

IDMSLBLS procedure listing

```

/* ----- LABELS -----
// DLBL dccat, 'idms.system.dccat',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,31
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dccatl, 'idms.system.dccatlod',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dccatx, 'idms.system.dccatx',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcdml, 'idms.system.ddldml',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,101
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclod, 'idms.system.ddldclod',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,21
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcllog, 'idms.system.ddldcllog',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,401
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcrun, 'idms.system.ddldcrun',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,68
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcscr, 'idms.system.ddldcscr',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,135
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dcmsg, 'idms.sysmsg.ddldcmsg',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dclscr, 'idms.sysloc.ddlocscr',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dirldb, 'idms.sysdir1.ddldml',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,201
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL dirllod, 'idms.sysdir1.ddldclod',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,2
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL empdemo, 'idms.empdemo1',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL insdemo, 'idms.insdemo1',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL orgdemo, 'idms.orgdemo1',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL emp1dem, 'idms.sqldemo.emp1demo',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,11
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL infodem, 'idms.sqldemo.infodemo',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL projdem, 'idms.projseq.projdemo',2099/365,DA
// EXTENT SYSnnn,nnnnnn,,ssss,6
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR

```

```

// DLBL      indxdem, 'idms.sqldemo.indxdemo', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 6
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      sysctl, 'idms.sysctl', 2099/365, SD
// EXTENT   SYSnnn, nnnnnn, , , ssss, 2
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      secdd, 'idms.sysuser.ddlsec', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 26
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      dictdb, 'idms.appldict.ddldml', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 51
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      dloddb, 'idms.appldict.ddldclod', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 51
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      sqldd, 'idms.syssql.ddlcat', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 101
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      sqlod, 'idms.syssql.ddlcatl', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 51
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      sqlxdd, 'idms.syssql.ddlcatx', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 26
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      asfdml, 'idms.asfdict.ddldml', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 201
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      asflod, 'idms.asfdict.asflod', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 401
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      asfdata, 'idms.asfdict.asfdata', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 201
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      ASFDEFN, 'idms.asfdict.asfdefn', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 101
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      j1jrn1, 'idms.j1jrn1', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 54
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      j2jrn1, 'idms.j2jrn1', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 54
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      j3jrn1, 'idms.j3jrn1', 2099/365, DA
// EXTENT   SYSnnn, nnnnnn, , , ssss, 54
// ASSGN    SYSnnn, DISK, VOL=nnnnnn, SHR
// DLBL      SYSIDMS, '#SYSIPT', 0, SD
/+
/*

```

<u>idmslib.sublib</u>	Name of the sublibrary within the library containing CA-IDMS modules
<u>user.sublib</u>	Name of the sublibrary within the library containing user modules
<u>idmslib</u>	Filename of the file containing CA-IDMS modules

<u>idms.library</u>	File-ID associated with the file containing CA-IDMS modules
<u>SYSnnn</u>	Logical unit of the volume for which the extent is effective
<u>nnnnnn</u>	Volume serial identifier of appropriate disk volume
<u>ssss</u>	Starting track (CKD) or block (FBA) of disk extent
dccat	Filename of the system dictionary catalog (DDL CAT) area
idms.system.dccat	File-ID of the system dictionary catalog (DDL CAT) area
dccatl	Filename of the system dictionary catalog load (DDL CATLOD) area
idms.system.dccatlod	File-ID of the system dictionary catalog load (DDL CATLOD) area
dccatx	Filename of the system dictionary catalog index (DDL CATX) area
idms.system.dccatx	File-ID of the system dictionary catalog index (DDL CATX) area
dcdml	Filename of the system dictionary definition (DDL DML) area
idms.system.ddldml	File-ID of the system dictionary definition (DDL DML) area
dclod	Filename of the system dictionary definition load (DDL DCLOD) area
idms.system.ddldclod	File-ID of the system dictionary definition load (DDL DCLOD) area
dcllog	Filename of the system log area (DDL DCLOG) area
idms.system.ddldcllog	File-ID of the system log (DDL DCLOG) area
dcrun	Filename of the system queue (DDL DCRUN) area
idms.system.ddldcrun	File-ID of the system queue (DDL DCRUN) area
dcscr	Filename of the system scratch (DDL DCSCR) area
idms.system.ddldcscr	File-ID of the system scratch (DDL DCSCR) area
dcmsg	Filename of the system message (DDL DCMSG) area
idms.sysmsg.ddldcmsg	File-ID of the system message (DDL DCMSG) area
dclscr	Filename of the local mode system scratch (DDL OCSCR) area

idms.sysloc.ddlocscr	File-ID of the local mode system scratch (DDLOCSCR) area
dirldb	Filename of the IDMSDIRL definition (DDLDM1) area
idms.sysdir1.ddldm1	File-ID of the IDMSDIRL definition (DDLDM1) area
dirllod	Filename of the IDMSDIRL definition load (DDLDCLOD) area
idms.sysdir1.dirllod	File-ID of the IDMSDIRL definition load (DDLDCLOD) area
empdemo	Filename of the EMPDEMO area
idms.empdemo1	File-ID of the EMPDEMO area
insdemo	Filename of the INSDEMO area
idms.insdemo1	File-ID of the INSDEMO area
orgdemo	Filename of the ORGDEMO area
idms.orgdemo1	File-ID of the ORGDEMO area
empldem	Filename of the EMPLDEMO area
idms.sqldemo.empldemo	File-ID of the EMPLDEMO area
infodem	Filename of the INFODEMO area
idms.sqldemo.infodem	File-ID of the INFODEMO area
projdem	Filename of the PROJDEMO area
idms.projseg.projdemo	File-ID of the PROJDEMO area
indxdem	Filename of the INDXDEMO area
idms.sqldemo.indxdemo	File-ID of the INDXDEMO area
sysctl	Filename of the SYSCTL file
idms.sysctl	File-ID of the SYSCTL file
secdd	Filename of the system user catalog (DDLSEC) area
idms.sysuser.ddlsec	File-ID of the system user catalog (DDLSEC) area
dictdb	Filename of the application dictionary definition area
idms.appldict.ddldm1	File-ID of the application dictionary definition (DDLDM1) area
dloddb	Filename of the application dictionary definition load area
idms.appldict.ddldclod	File-ID of the application dictionary definition load (DDLDCLOD) area
sqldd	Filename of the SQL catalog (DDLDCAT) area

idms.syssql.ddlcat	File-ID of the SQL catalog (DDL CAT) area
sqllod	Filename of the SQL catalog load (DDL CATL) area
idms.syssql.ddlcatl	File-ID of SQL catalog load (DDL CATL) area
sqlxdd	Filename of the SQL catalog index (DDL CATX) area
idms.syssql.ddlcatx	File-ID of the SQL catalog index (DDL CATX) area
asfdml	Filename of the asf dictionary definition (DDL DML) area
idms.asfdict.ddldml	File-ID of the asf dictionary definition (DDL DML) area
asflod	Filename of the asf dictionary definition load (ASF LOD) area
idms.asfdict.asflod	File-ID of the asf dictionary definition load (ASF LOD) area
asfdata	Filename of the asf data (ASF DATA) area
idms.asfdict.asfdata	File-ID of the asf data area (ASF DATA) area
ASFDEFN	Filename of the asf data definition (ASF DEFN) area
idms.asfdict.asfdefn	File-ID of the asf data definition area (ASF DEFN) area
j1jrn1	Filename of the first disk journal file
idms.j1jrn1	File-ID of the first disk journal file
j2jrn1	Filename of the second disk journal file
idms.j2jrn1	File-ID of the second disk journal file
j3jrn1	Filename of the third disk journal file
idms.j3jrn1	File-ID of the third disk journal file
SYSIDMS	Filename of the SYSIDMS parameter file

Index

Special Characters

#FREESTG macro E-29

#GETSTG macro E-29

#MOPT macro E-29

A

abnormal termination 5-30—5-31

addressed access

 addressed backward processing 3-6

 addressed direct processing 3-6

 addressed sequential processing 3-6

 general discussion 3-6

alternate index

 general discussion 2-10—2-11

 location mode options 2-10, 2-13

 record and set representations 2-10, 2-12—2-13

 schema definition example 2-11

application execution 5-30—5-31

application preparation 5-11—5-26

architecture 1-6—1-7, A-3—A-11

asynchronous processing 3-8

B

back-end module 1-7, A-7

backup and recovery 5-30

batch processing 5-14—5-18, 5-20—5-23, 5-30, A-4

C

CA-IDMS/DB

See also DML commands, DMCL, schema, subschema

 area considerations 2-14, 5-5

 central version 5-4, 5-5, 5-8, 5-11—5-16,
 5-19—5-21, 5-29, 5-30

 file considerations 2-14

 local mode 5-6, 5-9, 5-11, 5-16—5-18, 5-21—5-23,
 5-30

 location modes 2-4—2-5, 2-6, 2-8—2-9, 2-10, 2-13

 records and sets 2-4, 2-6, 2-8, 2-10, 2-12—2-13

CA-IDMS/VSAM Transparency

 abnormal termination 5-30—5-31

 application execution 5-30—5-31

 application preparation 5-11—5-26

 architecture 1-6—1-7, A-3—A-11

 backup and recovery 5-30

 command interface 5-27—5-29

CA-IDMS/VSAM Transparency (*continued*)

 database environment 1-6

 database preparation 5-4—5-10

 general discussion 1-8

 implementation 1-6

 initialization 5-28—5-29

 installation B-3—B-9

 integration with CA-IDMS/DB 1-4—1-5

 normal termination 5-30

 operating requirements 1-8

 overview 1-3—1-6

 parameters 5-23—5-26

 record access and processing 3-3—3-9

 return codes and messages D-3—D-19

 runtime operations 5-3—5-31

 shutdown procedures 5-31

 system commands 5-27—5-28

 system execution 5-27—5-29

 variable-length records C-3—C-4

chained request parameter lists 3-8

CICS

 SYSESVS VSE/ESA parameters G-3—G-6

CICS interface module 1-7, A-8

CICS processing 5-11—5-14, 5-19—5-20, 5-30, A-4,
 A-8—A-9, B-8, G-3—G-6

CICSOPTS B-8

command interface 5-27—5-29

concurrent access 3-8

control interval access 3-8

control tables

 compiler-directive statements 4-3, 4-5—4-6

 FMT 4-3, 4-7—4-16, G-4, G-6

 general discussion 1-7, A-10—A-11

 preparation of 4-3—4-22, 5-5

 TNT 4-3, 4-17—4-22

D

database preparation 5-4—5-10

DC/UCF 5-4

DMCL 5-4

DML commands 3-3, 3-4, 3-5, 3-6, 3-7

E

entry-sequenced data set

See ESDS

EODAD exit 3-8

error codes

 CLOSE D-22

 OPEN D-20

 REQUEST macro D-23

 TCLOSE D-22

ESDS

 general discussion 2-6—2-7

 location mode options 2-6, 2-13

 record and set representations 2-6, 2-12—2-13

 schema definition examples 2-7

ESVSALLO command F-4

ESVSMIGR 5-5—5-10

ESVSPVLR C-4

ESVSSPVT B-7

exception exit 3-9

EXCPAD exit 3-8

F

feedback codes D-20

file

 allocation F-3

FMT

 compiler JCL 4-13—4-16

 compiler messages D-4—D-7

 definition of 1-7, 4-3, A-10

 examples 4-12—4-13

 general discussion 4-7—4-16, 5-5, A-10

 syntax 4-7—4-10

 Usage 4-10—4-11

 use with ESVSMIGR 5-5

 variable-length record considerations C-4

 VSE/ESA CICS SYSESVS parameters G-4, G-6

front-end module 1-7, A-6

I

IDMS\$SVC B-7

IDMSINIT 5-5

IDMSINTC A-8, B-6

 OS/390 B-6

index component processing 3-8

initialization 5-28—5-29

installation B-5

 OS/390 B-4

 VSE/ESA B-7—B-9

ISAM interface 3-8

J

JCL (OS/390)

 application 5-11—5-18

 ESVSMIGR 5-5—5-7

 FMT compiler 4-13—4-15

 initialization 5-28

 installation B-4—B-5

 TNT compiler 4-19—4-20

JCL (VSE/ESA)

 application 5-11, 5-18—5-23

 ESVSMIGR 5-8—5-10

 FMT compiler 4-15—4-16

 initialization 5-28—5-29

 installation B-7

 TNT compiler 4-20—4-22

JRNAD exit 3-8

K

key-sequenced data set

See KSDS

keyed access

 general discussion 3-4—3-5

 keyed backward processing 3-5

 keyed direct processing 3-4

 keyed sequential processing 3-4

 keyed skip-sequential processing 3-5

KSDS

 general discussion 2-4—2-5

 location mode options 2-4—2-5, 2-13

 record and set representations 2-4, 2-12—2-13

 schema definition example 2-5

L

LERAD exit 3-8

location modes

 CALC 2-4, 2-6, 2-8, 2-10

 DIRECT 2-4, 2-6, 2-8

 VIA 2-4, 2-6, 2-8, 2-10

M

macros E-29

N

normal termination 5-30

O

operating requirements 1-8

P

PATH 2-10

See also alternate index

R

record access and processing 3-3—3-9

relative-record data set

See RRDS

relative-record number 2-8

request processing modules A-6—A-7

return codes and messages

command interface messages D-11—D-19

compiler messages D-4—D-10

general discussion D-3—D-19

reusable files 3-8

RRDS

general discussion 2-8—2-9

location mode options 2-8—2-9, 2-13

record and set representations 2-8, 2-12—2-13

schema definition example 2-9

run-time feedback codes D-20

runtime operations 5-3—5-31

S

schema

area and file definitions 2-14

definition summary 2-12—2-13

preparation of 5-4

VSAM file correspondences with 2-3—2-14

shared resources 3-8

shutdown procedures 5-31

shutdown procedures 5-31

subschema 5-4

SYNAD exit 3-8

synchronous processing 3-8

SYSESVS parameters G-3—G-6

system commands 5-27—5-28

system execution 5-27—5-29

system services manager 1-7, A-5

T

TNT

compiler JCL 4-19—4-22

compiler messages D-7—D-10

TNT (*continued*)

definition of 1-7, 4-3

examples 4-18

general discussion 4-17—4-22, 5-5, A-8,
A-10—A-11

syntax 4-17—4-18

usage examples 4-18

TSO considerations F-4

U

UPGRADE set 2-10

user exit

COBOL control block E-5

user exits 3-8, E-3

after E-3

assembler E-17

Assembler control block DSECT

—#EVEXBDS E-18

assembler macros E-29

assembler sample E-22

assembler template E-21

before E-3

COBOL E-4

COBOL control block fields E-7

COBOL sample E-11

user security verification exit 3-9

V

variable-length records 2-3, 3-8, C-3—C-4

VSAM

data migration to CA-IDMS/DB 5-5—5-10

DELETE command 3-7

error message area 3-8

GET 3-4, 3-5

OS/390 exits 3-8, 3-9

POINT 3-4, 3-5

processing options 3-3—3-9

READ BY KEY command 3-7

READ NEXT command 3-7

READ PREVIOUS command 3-7

REWRITE command 3-7

START/STARTBR command 3-7

VSE/ESA CICS SYSESVS parameters G-3—G-6

VSE/ESA exits 3-8, 3-9

WRITE command 3-7

