CA IDMS - 19.0
Using VSAM Transparency

Date: 16-Jan-2018
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2017 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
# Table of Contents

**Introduction to CA IDMS VSAM Transparency** .................................................. 9

Data Access Method Conversion Process ......................................................................................... 9
Integration with CA IDMS/DB ........................................................................................................... 10
Implementation .................................................................................................................................. 11
General Architecture .......................................................................................................................... 12
Operating Requirements ...................................................................................................................... 13

**CA IDMS/DB Schema and VSAM File Correspondences** ........................................... 13

Key-Sequenced Data Set .................................................................................................................. 14
  Record and Set Representations ........................................................................................................... 15
  Location Mode Options ....................................................................................................................... 15
  Example of Schema Definition ............................................................................................................ 15
Entry-Sequenced Data Set .................................................................................................................. 16
  Record and Set Representations ........................................................................................................... 16
  Location Mode Options ....................................................................................................................... 16
  Examples of Schema Definition ......................................................................................................... 17
Relative-Record Data Set ..................................................................................................................... 17
  Record and Set Representations ........................................................................................................... 18
  Location Mode Options ....................................................................................................................... 18
  Example of Schema Definition ......................................................................................................... 19
Alternate Index .................................................................................................................................... 19
  Record and Set Representations ........................................................................................................... 19
  Location Mode Options ....................................................................................................................... 20
  Example of Schema Definition ......................................................................................................... 20

Schema Definition Summary ............................................................................................................. 20
  Schema Definitions for VSAM Data Structures .................................................................................. 21
Area and File Considerations .............................................................................................................. 22

**Record Access and Processing** ................................................................................................. 23

Keyed Access ..................................................................................................................................... 23
  Keyed Direct Processing ....................................................................................................................... 24
  Keyed Sequential Processing ................................................................................................................. 24
  Keyed Skip-Sequential Processing ....................................................................................................... 25
Keyed Backward Processing .............................................................................................................. 25
Addressed Access ............................................................................................................................. 25
Addressed Direct Processing .............................................................................................................. 26
Addressed Sequential Processing ....................................................................................................... 26
Addressed Backward Processing ......................................................................................................... 26
DML Correspondences To VSAM Commands ....................................................................................... 27
Additional VSAM Options .................................................................................................................. 27

Preparing Control Information ........................................................................................................ 28
Preparing Control Information Overview .......................................................................................... 29
Compiler-Directive Statements 3 ......................................................................................................... 30
Example ............................................................................................................................................... 32
File Management Table ...................................................................................................................... 32
Usage .................................................................................................................................................. 35
Examples ............................................................................................................................................. 36
FMT JCL .............................................................................................................................................. 37
Transaction Name Table ...................................................................................................................... 39
Examples ............................................................................................................................................. 40
TNT JCL .................................................................................................................................................. 41

Runtime Operations .......................................................................................................................... 43
Database Preparation ............................................................................................................................ 44
Step 1 Install and Prepare the DC/UCF System .................................................................................. 44
Step 2 Prepare and Install the CA IDMS/DB Database ...................................................................... 44
Step 3 Create and Compile the Control Tables .................................................................................. 45
Step 4 Migrate Data to the CA IDMS/DB Database ......................................................................... 45
  Sample migration utility JCL -- z/OS ............................................................................................... 46
  Sample migration utility JCL -- z/VSE ............................................................................................ 47
Application Preparation ....................................................................................................................... 49
Step 1 Modify the Application JCL ...................................................................................................... 49
  Modify the application JCL -- z/OS ................................................................................................. 50
  Modify the application JCL -- z/VSE .............................................................................................. 54
  CA IDMS VSAM Transparency parameters .................................................................................... 58
Step 2 Modify and Recompile the Application Program ................................................................... 60
System Execution ................................................................................................................................. 60
Step 1 Run the CA IDMS VSAM Transparency Command Interface ............................................... 60
  CA IDMS VSAM Transparency system commands ....................................................................... 61
  CA IDMS VSAM Transparency initialization -- z/OS .................................................................... 61
  CA IDMS VSAM Transparency initialization -- z/VSE .................................................................. 62
# Table of Contents

**Step 2 Bring Up the DC/UCF System** ................................................................. 63
  Application Execution ...................................................................................... 63
  Normal Termination .......................................................................................... 63
  Abnormal Termination ...................................................................................... 64
  CA IDMS VSAM Transparency shutdown procedures -- z/OS ....................... 64

**CA IDMS VSAM Transparency Architecture** ................................................. 65
  System Services Manager .................................................................................. 65
  CA IDMS VSAM Transparency Architecture -- Batch Processing ................. 65
  CA IDMS VSAM Transparency Architecture -- CICS Processing .................. 66
  Request Processing Modules .......................................................................... 68
    The CA IDMS VSAM Transparency Front End .................................................. 68
    The CA IDMS VSAM Transparency Back End ............................................... 69
  Control Tables .................................................................................................. 70
    File Management Table .............................................................................. 70
    Transaction Name Table ............................................................................ 71

**CA IDMS VSAM Transparency Installation** ................................................... 71
  z/OS Installation ............................................................................................. 71
    Installing CA IDMS VSAM Transparency in the CICS Environment ............ 73
  z/VSE Installation ........................................................................................... 74
    Installing CA IDMS VSAM Transparency in the CICS Environment ............ 75

**Variable-Length Record Considerations** ...................................................... 76

**CA IDMS VSAM Transparency User Exits** ................................................... 77
  COBOL User Exit Program ............................................................................. 78
    Program Requirements .................................................................................. 78
    COBOL User Exit Control Block .................................................................. 80
    Sample COBOL User Exit ......................................................................... 85
  Assembler User Exit Program ........................................................................ 90
    Program Requirements ................................................................................ 90
    Assembler User Exit Control Block ............................................................... 92
    Assembler User Exit Template .................................................................... 95
    Sample Assembler User Exit ..................................................................... 96
  Macros ............................................................................................................. 102
Using VSAM Transparency

What is CA IDMS VSAM Transparency?

CA IDMS VSAM Transparency is a program interface that allows VSAM-based applications to access and update an 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.

This section provides introductory information on the following:

- How CA IDMS VSAM Transparency converts and integrates data to an 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

For more information, see the following topics:

- Introduction to CA IDMS VSAM Transparency (see page 9)
- CA IDMS/DB Schema and VSAM File Correspondences (see page 13)
- Record Access and Processing (see page 23)
- Preparing Control Information (see page 28)
- Runtime Operations (see page 43)
- CA IDMS VSAM Transparency Architecture (see page 65)
- CA IDMS VSAM Transparency Installation (see page 71)
- Variable-Length Record Considerations (see page 76)
- CA IDMS VSAM Transparency User Exits (see page 77)
- TSO File Allocation with CA IDMS VSAM Transparency (see page 103)
Introduction to CA IDMS VSAM Transparency

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)

For more information, see the following CA IDMS VSAM Transparency topics:

- Data Access Method Conversion Process (see page 9)
- Integration with CA IDMS/DB (see page 10)
- Implementation (see page 11)
- General Architecture (see page 12)
- Operating Requirements (see page 13)

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.

Integration with CA IDMS/DB

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 an CA IDMS/DB database. CA IDMS VSAM Transparency acts as a translator between VSAM and CA IDMS/DB, allowing VSAM application programs to access an CA IDMS/DB database.

**Integration with CA IDMS/DB**

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.

**Implementation**

To implement CA IDMS VSAM Transparency, follow these steps:
1. **Prepare the database** by following the procedures detailed in CA IDMS Database Administration Section.

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 .

3. **Prepare the control tables** that will be used by CA IDMS VSAM Transparency at runtime. Instructions on preparing control tables are presented in .

4. **Modify the application JCL.** Sample JCL is presented in Section5, Runtime Operations.

---

**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 an 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 an 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.

More information:

For more information on preparing control table information see Section4, Preparing Control Information.

For more detailed information on CA IDMS VSAM Transparency architecture, refer to Appendix A, CA IDMS VSAM Transparency Architecture.
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 z/OS in the Common System area (CSA). This release of CA IDMS supports z/OS 1.1 and above. However, we will always refer to z/OS in this section.
    - For z/VSE, 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 an CA IDMS/DB run unit.

- **CA 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.

CA IDMS/DB Schema and VSAM File Correspondences

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 an 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 an 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.

The remainder of this section 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 Length Record Considerations.

For more information, see the following topics:

- Key-Sequenced Data Set (see page 14)
- Entry-Sequenced Data Set (see page 16)
- Relative-Record Data Set (see page 17)
- Alternate Index (see page 19)
- Schema Definition Summary (see page 20)
- Area and File Considerations (see page 22)

## Key-Sequenced Data Set

### Contents

- Record and Set Representations (see page 15)
- Location Mode Options (see page 15)
- Example of Schema Definition (see page 15)

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.

Record and Set Representations

In an 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 an 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

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.

Example of Schema Definition

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

Contents
- Record and Set Representations (see page 16)
- Location Mode Options (see page 16)
- Examples of Schema Definition (see page 17)

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.

Record and Set Representations

In an 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 an 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

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.

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.
```

Relative-Record Data Set

Contents

- Record and Set Representations (see page 18)
- Location Mode Options (see page 18)
- Example of Schema Definition (see page 19)
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.

Record and Set Representations

In an 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 an 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.

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.

**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.
```

**Alternate Index**

**Contents**
- Record and Set Representations (see page 19)
- Location Mode Options (see page 20)
- Example of Schema Definition (see page 20)

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.

**Record and Set Representations**

In an 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.

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.

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 ORDER IS SO
  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 ORDER IS SORTED
  MODE IS INDEX
  OWNER IS SYSTEM
  MEMBER IS DEPARTMENT
  ASCENDING KEY IS DEPARTMENT-NAME
  DUPLICATES LAST.
```

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.
Schema Definitions for VSAM Data Structures

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

<table>
<thead>
<tr>
<th>CA IDMS/DB schema definitions</th>
<th>KSDS data structure</th>
<th>ESDS data structure</th>
<th>RRDS data structure</th>
<th>Alternate index data structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSDS</td>
<td>KSDS- primekey-NDX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>I MA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASC prime-key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESDS</td>
<td>ESDS- record-NDX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>I MA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LAST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RRDS</td>
<td>RRDS- control-NDX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>I MA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASC control-field</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alternate index</td>
<td>KSDS- primekey-NDX</td>
<td>KSDS- altkey-NDX</td>
<td>ESDS- record-NDX</td>
<td>ESDS- altkey-NDX</td>
</tr>
<tr>
<td></td>
<td>I MA</td>
<td>I OA</td>
<td>I MA</td>
<td>I OA</td>
</tr>
<tr>
<td></td>
<td>ASC prime-key</td>
<td>ASC alternate-key</td>
<td>LAST</td>
<td>ASC alternate-key</td>
</tr>
</tbody>
</table>

Schema Definition Summary

Schema Definition Summary (2)
<table>
<thead>
<tr>
<th>CA IDMS/DB schema definitions</th>
<th>KSDS data structure</th>
<th>ESDS data structure</th>
<th>RRDS data structure</th>
<th>Alternate index data structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record type and set member</td>
<td>KSDS record definition</td>
<td>ESDS record definition</td>
<td>RRDS record definition</td>
<td>System-owned index</td>
</tr>
<tr>
<td>Set type</td>
<td>System-owned index</td>
<td>System-owned index</td>
<td>System-owned index</td>
<td>System-owned index</td>
</tr>
<tr>
<td>Set order</td>
<td>Sorted on prime key</td>
<td>Sorted on db-key or LAST</td>
<td>Sorted on the added control field</td>
<td>Sorted on the alternate key</td>
</tr>
<tr>
<td>Duplicates</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed or LAST</td>
</tr>
<tr>
<td>Location mode</td>
<td>VIA indexed set or other set CALC prime key or other field DIRECT</td>
<td>VIA indexed set or other set CALC other field DIRECT</td>
<td>VIA indexed set or other set CALC control field or other field DIRECT</td>
<td>VIA alternate key CALC alternate key</td>
</tr>
</tbody>
</table>

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 an CA IDMS/DB file, and assign the file to an external ddname (z/OS) or filename (z/VSE).

More information:

Instructions for defining areas and files are presented in CA IDMS Database Administration Section.

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 (z/OS) or filename (z/VSE) 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:

  ```cobol
  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

More information:

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

Record Access and Processing

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

For more information, see the following topics:
- Keyed Access (see page 23)
- Addressed Access (see page 25)
- DML Correspondences To VSAM Commands (see page 27)
- Additional VSAM Options (see page 27)

Keyed Access

Contents
- Keyed Direct Processing (see page 24)
- Keyed Sequential Processing (see page 24)
- Keyed Skip-Sequential Processing (see page 25)
- Keyed Backward Processing (see page 25)

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.

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.

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.
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.

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.

Addressed Access

Contents

- Addressed Direct Processing (see page 26)
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.

**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.

**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.

**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.

### 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.

<table>
<thead>
<tr>
<th>VSAM Command</th>
<th>CA IDMS/DB DML Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ BY KEY</td>
<td>OBTAIN WITHIN SET USING SORTKEY</td>
</tr>
<tr>
<td>START/STARTBR</td>
<td>FIND WITHIN SET USING SORTKEY</td>
</tr>
<tr>
<td>READ NEXT</td>
<td>OBTAIN FIRST/NEXT WITHIN SET</td>
</tr>
<tr>
<td>READ PREVIOUS</td>
<td>OBTAIN LAST/PRIOR WITHIN SET</td>
</tr>
<tr>
<td>WRITE</td>
<td>STORE</td>
</tr>
<tr>
<td>REWRITE</td>
<td>MODIFY</td>
</tr>
<tr>
<td>DELETE</td>
<td>ERASE</td>
</tr>
</tbody>
</table>

### 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 z/OS 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 z/VSE 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

Preparing Control Information

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.

This section 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.

For more information, see the following topics:

- Preparing Control Information Overview (see page 29)
- Compiler-Directive Statements 3 (see page 30)
- File Management Table (see page 32)
- Transaction Name Table (see page 39)

Preventing Control Information Overview

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.
Compiler-Directive Statements 3

Contents

- Example (see page 32)

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

```
CORE size=nnnnn

ICTL=(start-column,end-column)
```
Parameters

CORE size=nnnnn

Specifies the amount of storage the compiler is to acquire (by a GETMAIN under z/OS or by a GETVIS or COMREG under z/VSE) 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.

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 *

File Management Table

Contents

- Usage (see page 35)
- Examples (see page 36)
- FMT JCL (see page 37)

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 z/VSE CA IDMS/DB 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 later in this section for VSAM data set definitions, CA IDMS schema definitions and corresponding FMT compiler statements. See later in this section for z/VSE CICS SYSESVS file management parameters.

Syntax

Required for KSDS and a PATH Only
**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.
  More information:
  For specific VSAM cluster statements that indicate the file type, see Usage (see page 35)

- **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:
  
  $((\text{String number or number of data buffers specified in the ACB}) \times \text{maximum VSAM record length})$

- 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.
**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.

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

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

<table>
<thead>
<tr>
<th>FMT statements</th>
<th>LISTCAT description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMT NAME IS EMPFMT</td>
<td>INDEXED</td>
</tr>
<tr>
<td>FILE TYPE IS KSDS</td>
<td>KEYLEN------------44</td>
</tr>
<tr>
<td>KEY LENGTH IS 44</td>
<td>RKP---------------0</td>
</tr>
<tr>
<td>KEY POSITION IS 0</td>
<td>MAXLRECL-----------805</td>
</tr>
<tr>
<td>VSAM RECORD LENGTH IS 805</td>
<td>FILE IS NOT REUSABLE</td>
</tr>
<tr>
<td>RECORD NAME IS EMPLOYEE</td>
<td>SET NAME IS EMP-NDX</td>
</tr>
</tbody>
</table>

### VSAM ESDS

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

<table>
<thead>
<tr>
<th>FMT statements</th>
<th>LISTCAT description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMT NAME IS DEPFMT</td>
<td>NONINDEXED</td>
</tr>
<tr>
<td>FILE TYPE IS ESDS</td>
<td>MAXLRECL-----------17</td>
</tr>
<tr>
<td>VSAM RECORD LENGTH IS 17</td>
<td>FILE IS REUSABLE</td>
</tr>
<tr>
<td>RECORD NAME IS DEPARTMENT</td>
<td>SET NAME IS DEPT-NDX</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>FMT statements</th>
<th>LISTCAT description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMT NAME IS EXPFMT</td>
<td></td>
</tr>
<tr>
<td>FILE TYPE IS RRDS</td>
<td>NUMBERED</td>
</tr>
<tr>
<td>RELATIVE RECORD NUMBER POSITION IS 9</td>
<td></td>
</tr>
<tr>
<td>VSAM RECORD LENGTH IS 9</td>
<td>MAXLRECL-------------9</td>
</tr>
<tr>
<td>RECORD NAME IS EXPERTISE</td>
<td></td>
</tr>
<tr>
<td>SET NAME IS EXP-NDX</td>
<td></td>
</tr>
</tbody>
</table>

**VSAM PATH**

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

<table>
<thead>
<tr>
<th>FMT statements</th>
<th>LISTCAT description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMT NAME IS MGTFMT</td>
<td></td>
</tr>
<tr>
<td>FILE TYPE IS PATH</td>
<td></td>
</tr>
<tr>
<td>BASE CLUSTER IS KSDS</td>
<td></td>
</tr>
<tr>
<td>KEY LENGTH IS 8</td>
<td>KEYLEN---------------8</td>
</tr>
<tr>
<td>KEY POSITION IS 5</td>
<td>RKP-------------------5</td>
</tr>
<tr>
<td>VSAM RECORD LENGTH IS 17</td>
<td>MAXLRECL-------------17</td>
</tr>
<tr>
<td>FILE IS REUSABLE</td>
<td>REUSE</td>
</tr>
<tr>
<td>RECORD NAME IS EMPLOYEE</td>
<td></td>
</tr>
<tr>
<td>SET NAME IS MANAGES-NDX</td>
<td></td>
</tr>
</tbody>
</table>

**FMT JCL**

The JCL used to compile, assemble, and link edit an FMT is presented below.

**FMT (z/OS)**

```bash
//****************************************************************************
//*                                                                         *
//*        COMPILE FMT                                                    *
//*                                                                         *
******************************************************************************
//FMTCC      EXEC  PGM=ESVSFMTC,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=.,DISP=(NEW,PASS)
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=4560),
//            SPACE=(CYL,(1,1)),
//            UNIT=disk
```
CA IDMS - 19.0

//SYSIDMS DD *Input SYSIDMS parameters, as required//SIPT DD *Insert FMT input statements here/*
//******************************************************************************
/*
/* ASSEMBLE COMPILER OUTPUT
/*
/*****************************************************************************/
//ASM EXEC PGM=ASMA90,PARM='NLOAD,DECK',COND=(4,LT,S1),
 // REGION=512K
//SYSPRINT DD DUMMY
//SYSU1 DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSU2 DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSPUNCH DD DSN=.&.fmtobj.,UNIT=disk,DISP=(NEW,PASS),
 // SPACE=(80,(400,40))
//SYIN 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 SYSLIN=A
//SYSLIN DD DSN=.&.fmtobj.,DISP=(OLD,DELETE)
//SYUT1 DD UNIT=disk,SPACE=(TRK,(20,5))
//SYLMOD DD DSN=userid.loadlib(fmtname),DISP=SHR

idms.dba. Data set name of the CA IDMS/DB load library containing CA IDMS DMCL and database
loadlib.

idms. Data set name of the CA IDMS/DB load library containing ESVSFMT.
loadlib.

.&.&fmt. Temporary data set that contains output from the compile step.
disk. Symbolic device type for the disk file.

&. Temporary data set that contains output from the assembly step.
&fmtobj.

user. Data set name of the user load library where the FMT load module will be placed.
loadlib.

fmtname Name of the FMT load module.

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 CA IDMS Common
Facilities Section.

FMT (z/VSE)

******************************************************************************
/*
/* COMPILE FMT
/*
******************************************************************************
/* EXEC PROC=IDMSLBLS
/* DLBL idmslib,'idms.lib'
/* EXTENT ,xxxxxx
/* LIBDEF PHASE, SEARCH=idmslib.sublib
/* DLBL idmspch, 'work', SD
/* EXTENT SYS020,nnnnn,,ssss,1111// ASSGN SYS020,disk
/* EXEC ESVSFMTInput SYSIDMS parameters, as required/Insert FMT input statements
here/*
******************************************************************************
/*
Transaction Name Table

Contents

- Examples (see page 40)
- TNT JCL (see page 41)

Purpose

The transaction name table (CICS only) specifies correspondences between a CICS transaction and the CA IDMS/DB subschema that the transaction will use. A TNT is optional if all CA IDMS VSAM Transparency CICS transactions use the same subschema. It is required if different CA IDMS VSAM Transparency CICS transactions use different subschemas.

If no TNT is specified, CA IDMS VSAM Transparency uses the subschema named in the JCL, or from the SYSESVS parameters (under z/VSE CICS only).
More information:

For more information on z/VSE CICS SYSESVS parameters, see z/VSE CICS SYSESVS Parameters (see page 105)

Syntax

- **TNT NAME is transaction-name-table-name**
- **TRANSACTION NAME is transaction-name**
- **SUBschema NAME is subschema-name**
- **NODENAME is nodename**
- **DBNAME is database-name**

Parameters

TNT NAME is *transaction-name-table-name*

A user-defined TNT name of 1 to 8 characters.

- **TRANSACTION NAME is transaction-name**
  Specifies a 1- to 8-character transaction code or transaction name. There can be several occurrences of this statement for each TNT. The transaction names can be obtained from a CICS program control table (PCT) listing.
  If the transaction is not specified in the TNT, CA IDMS VSAM Transparency uses the subschema specified in the application JCL, or from the SYSESVS parameters (under z/VSE CICS only).
  More information:
  For more information on z/VSE CICS SYSESVS parameters, see z/VSE CICS SYSESVS Parameters (see page 105) if no default has been specified, the transaction is not processed.

- **SUBschema NAME is subschema-name**
  Specifies a 1- to 8-character subschema name that will be used by CA IDMS VSAM Transparency to process the named transaction.

- **NODENAME is nodename**
  Specifies the node name that contains the database to be accessed by CA IDMS VSAM Transparency for this transaction. Node name can be from 1 to 8 characters.

- **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.

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 z/VSE CICS only).

More information:

For more information on z/VSE CICS SYSESVS parameters, see

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 z/VSE CICS only).

More information:

For more information on z/VSE CICS SYSESVS parameters, see num=G.z/VSE CICS SYSESVS Parameters (see page 105)

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.

TNT JCL

The JCL used to compile, assemble, and link edit a TNT is presented below.

z/OS (TNT)

/********************************************
/ * COMPILER JCL for 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=F8,LRECL=80,BLKSIZE=4560),
//         SPACE=(CYL,(1,1)),
//         UNIT=disk
//SYSIDMS DD *Input SYSIDMS parameters, as required//SYSIPT DD *Insert TNT
input statements here*/
/********************************************
/ * ASSEMBLER JCL for TNT                    *
/ ********************************************

//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
//****************************
//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

idms.dba. Data set name of the CA IDMS/DB load library containing CA IDMS DMCL and database loadlib
loadlib
idms. Data set name of the CA IDMS/DB load library containing ESVSTNTC loadlib
.&.&tnt. Temporary data set that contains output from the compile step
disk
.&.&tntobj. Temporary data set that contains output from the assembly step
user. Data set name of the user load library where the TNT load module will be placed loadlib
tntname Name of the TNT load module
SYSIDMS DDname of the parameter file provided by CA IDMS to specify runtime directives and
operating system-dependent parameters. A complete description of the SYSIDMS parameter file, see CA IDMS Common Facilities Section.

z/VSE (TNT)

********************************************************************************
* * COMPILe TNT
* *
********************************************************************************
//EXEC PROc=IDMSLBLS
//DLBL idmslib,'idms.lib'
//EXTENT ,xxxxxx
//LIBDEF PHASE, SEARCH=idmslib sublic
//DLBL IDMSPCH,'work',,SD
//EXTENT SYS020,nnnnnn,ssss,lli/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,lli ASSGN SYSIPT,DISK
//OPTION CATAL
//PHASE tntname,*
//EXEC ASMA90
//CLOSE SYSIPT,SYSRDR
### Runtime Operations

**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, see *CA IDMS Database Administration Section*.

This section discusses the operational considerations that are specific to CA IDMS VSAM Transparency and are presented as follows:

- Database Preparation (see page 44)
- Application Preparation (see page 49)
- System Execution (see page 60)
- Application Execution (see page 63)
Database Preparation

Contents

- Step 1 Install and Prepare the DC/UCF System (see page 44)
- Step 2 Prepare and Install the CA IDMS/DB Database (see page 44)
- Step 3 Create and Compile the Control Tables (see page 45)
- Step 4 Migrate Data to the CA IDMS/DB Database (see page 45)
  - Sample migration utility JCL -- z/OS (see page 46)
  - Sample migration utility JCL -- z/VSE (see page 47)

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.

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.

More information:

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

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 CA IDMS Database Administration Section.

- **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 Section.
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 Section.

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.

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.

More information:

For instructions on preparing and compiling the control tables can be found in Section 4, Preparing Control Information.

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 an 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 Section) 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 Preparing Control Information (see page 28), for instructions.

Sample migration utility JCL -- z/OS

The z/OS 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 (z/OS)

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

Insert SYSIDMS parameters, as appropriate

```
//STEPCONV EXEC PGM=ESVSMIGR,REGION=1024K
//STEPLIB DD DSN=SYSIDMS,DISP=SHR
```

Local Mode ESVSMIGR (z/OS)

```
//STEPCONV EXEC PGM=ESVSMIGR,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// dcmdl DD DSN=idms.sysctl,DISP=SHR
// dclod DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
// dcmlog DD DSN=idms.sysmsg.ddldclog,DISP=SHR
// SYSLST DD DUMMY
// userdb DD DSN=user.userdb,DISP=SHR
Additional DD statements as required
```

**Note:**

- **idms.dba.loadlib**: Data set name of the load library containing DMCL and database name table load modules
- **idms.loadlib**: Data set name of the load library containing the CA IDMS executable modules
- **idms.sysctl**: Data set name of the CA IDMS/DB SYSCTL file
- **dcmlog**: DDname of the system message (DDLDCMSG) area
- **idms.sysmsg.ddldcmsg**: Data set name of the system message (DDLDCMSG) area
- **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
- **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 CA IDMS Common Facilities Section.
Sample migration utility JCL -- z/VSE

The z/VSE 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 (z/VSE)
IDMS Name of the procedure provided at installation containing the file definitions for CA IDMS
LRLS dictionaries, databases, SYSIDMS, parameter file and other files.

xxxxx Volume serial number of disk unit

idmsl Filename of the CA IDMS Library

ib File-id of the CA IDMS Library

vsam File-id of the VSAM file to be migrated

file

fmtm Name of the file management table for the file to be migrated

ame

ssna Subschema name

me

nnnn Size of the buffer that CA IDMS VSAM Transparency will use for communication between the
nnn 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
MS system-dependent parameters.

For a complete description of the SYSIDMS parameter file, see CA IDMS Common Facilities
Section.

Local Mode ESVSMIGR (z/VSE)

// EXEC PROC=IDMSLBLS
// ASSGN  SYSnnn, IGN
// DLBL  idmslib,'idms.lib'
// EXTENT ,xxxxxx
// LIBDEF PHASE, SEARCH=idmslib.sublib
// DLBL  userdb,'user.userdb',,SD
// EXTENT ,xxxxxx
// DLBL  OUTPUT,'FMT=fmtname',,VSAM
// DLBL  ESVS,'SUBSCHEMA=ssname,RBUFSZ=nnnnn',,VSAM
// ASSGN  SYSLSLT,PRINTER
// EXEC  ESVSMIGR

Insert SYSIDMS parameters, as required/

IDMSL Name of the procedure provided at installation containing the file definitions for CA IDMS
LRLS dictionaries, databases, SYSIDMS, parameter file and other files.

nnn Logical unit assignment for the SYSJRNL in the DMCL

xxxxxx Volume serial number of disk unit
### Application Preparation

**Contents**

- Step 1 Modify the Application JCL (see page 49)
  - Modify the application JCL -- z/OS (see page 50)
  - Modify the application JCL -- z/VSE (see page 54)
- CA IDMS VSAM Transparency parameters (see page 58)
- Step 2 Modify and Recompile the Application Program (see page 60)

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.

### Step 1 Modify the Application JCL

**Two DD or DLBL Statements for Each File**

Each ddname (z/OS) or filename (z/VSE) 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 z/OS and z/VSE are presented below.

Modify the application JCL -- z/OS

SUBSYS Parameter

In z/OS, 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 (z/OS)

//ESVSCICS EXEC PGM=DFHSIP
//STEPLIB DD DSN=cics.system.loadlib1,DISP=SHR
//DFHRPL DD DSN=cics.system.loadlib2,DISP=SHR
// DD DSN=idsm.dba.loadlib,DISP=SHR
// DD DSN=idsm.loadlib,DISP=SHR
// DD DSN=idsm.sysmsg.loadlib1,DISP=SHR
// ddmsg DSN=idsm.sysmsg.ddldcmsg,DISP=SHR
//sysctl DD DSN=cics.system.sysctl,DISP=SHR
//userdd DD SUBSYS=(ESVS,’FMT=fmtname’,’SUBSCHEMA=ssname’
// ‘MODE=CICS’,’RBUFFSZ=nnnnn’,’TNT=tntname’
// ‘CWADISP=disp’)
// DD DSN=dataset-name,DISP=SHR
//SYSIDMS DD *

Insert SYSIDMS parameters, as appropriate

<table>
<thead>
<tr>
<th>user.loadlib</th>
<th>Data set name of the user load library</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdd</td>
<td>DDname of the CA IDMS VSAM Transparency file, as it appears in the application</td>
</tr>
<tr>
<td></td>
<td>Names of the CICS system load libraries</td>
</tr>
</tbody>
</table>
More information:

For more information on these and additional CA IDMS VSAM Transparency parameters, refer to 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:

```jcl
//ESVSCICS EXEC PGM=DFHSIP  
//STELIB DD DSN=CICS.SYSTEM.LOADLIB1, DISP=SHR  
//DFHRLPL DD DSN=CICS.SYSTEM.LOADLIB, DISP=SHR  
// DSN=CICS.SYSTEM.LOADLIB2, DISP=SHR  
// DSN=IDMS.DBA.LOADLIB, DISP=SHR  
// DSN=IDMS.LOADLIB, DISP=SHR  
// DSN=USER.LOADLIB, DISP=SHR  
//DD DSN=IDMS.SYSMSG.DDLDCMSG, DISP=SHR  
//SYSDM DD DSN=USER.SYSCON, DISP=SHR  
//SYSDM DD *  
//EMPFILE DD SUBSY=(ESVS,'FMT=EMPZMT', 'SUBSCHMA=EMPSS01', 'MODE=CICS', 'RBUFZ=500', 'TNT=CICSTNT', 'CWADISP=16'),
```
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 (z/OS)

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

Insert SYSIDMS parameters, as appropriate

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userpgm</td>
<td>Application program name</td>
</tr>
<tr>
<td>user.loadlib</td>
<td>Data set name of the user load library</td>
</tr>
<tr>
<td>cics.system.</td>
<td>Names of the CICS system load libraries</td>
</tr>
<tr>
<td>loadlib1,</td>
<td></td>
</tr>
<tr>
<td>loadlib2</td>
<td></td>
</tr>
<tr>
<td>idms.sysmsg.</td>
<td>Data set name of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>dldcmsg</td>
<td></td>
</tr>
<tr>
<td>idms.dba.</td>
<td>Data set name of the load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td>loadlib</td>
<td></td>
</tr>
<tr>
<td>idms.loadlib</td>
<td>Data set name of the load library containing the CA IDMS executable system modules</td>
</tr>
<tr>
<td>dcmsg</td>
<td>DDname of the system message (DDLDCMSG) area</td>
</tr>
<tr>
<td>sysctl</td>
<td>DDname of the user SYSCTL file</td>
</tr>
<tr>
<td>user.sysctl</td>
<td>Data set name of the user SYSCTL file</td>
</tr>
<tr>
<td>userdd</td>
<td>DDname of the CA IDMS Transparency file, as it appears in the application</td>
</tr>
<tr>
<td>fmtname</td>
<td>Name of the file management table that defines the escaped file</td>
</tr>
<tr>
<td>ssname</td>
<td>Subschema name</td>
</tr>
<tr>
<td>modetype</td>
<td>Optional indicator of BATCH or CICS mode; if not specified, the default is BATCH</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Buffer size</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>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 CA IDMS Common Facilities Section.</td>
</tr>
</tbody>
</table>

For more information on these and additional CA IDMS VSAM Transparency parameters, refer to 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
//STELIB DD      DSN=user.loadlib,DISP=SHR
//DD      DSN=idosms.dba.loadlib,DISP=SHR
//SYSCTL DD      DSN=idosms.sysctl,DISP=SHR
//DEPFILE DD     SUBSYS=(ESVS,'FMT=EMPFMT','SUBSCHEMA=EMPSS01,'
//                  'RBUFZS=nnnnn'
//ddmsg       DDDSN=idosm.sysmsg.ddldcmsg,DISP=SHR
//SYSDM DD     *
```

**Local Mode (z/OS)**

```
//userpgm EXEC   PGM=userpgm//STELIB DD      DSN=user.loadlib,DISP=SHR
//DD      DSN=idosms.dba.loadlib,DISP=SHR
//ddcml DD      DSN=idosms.system.ddldml,DISP=SHR
//ddclo DD      DSN=idosms.system.ddldclog,DISP=SHR
//SYSSJRNL DD    DSN=idosms.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
//SYSIDMS DD     *
//DD      DSN=idosm.sysmsg.ddldcmsg,DISP=SHR
```

Insert SYSIDMS parameters, as appropriate

<table>
<thead>
<tr>
<th>User Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userpgm</td>
<td>Application program name</td>
</tr>
<tr>
<td>user.loadlib</td>
<td>Data set name of the user load library</td>
</tr>
<tr>
<td>idms.dba.loadlib</td>
<td>Data set name of the load library containing the CA IDMS DMCL and database name table load modules</td>
</tr>
<tr>
<td>idms.loadlib</td>
<td>Data set name of the load library containing the CA IDMS system executable modules</td>
</tr>
<tr>
<td>dcdml</td>
<td>DDname of the system dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>idms.system.ddldml</td>
<td>Data set name of the system dictionary definition (DDLDML) area</td>
</tr>
<tr>
<td>dclod</td>
<td>DDname of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>idms.system.ddldclod</td>
<td>Data set name of the system dictionary definition load (DDLDCLOD) area</td>
</tr>
<tr>
<td>dcllog</td>
<td>DDname of the system log (DDLDCLOG) area</td>
</tr>
<tr>
<td>idms.system.ddldclog</td>
<td>Data set name of the system log (DDLDCLOG) area</td>
</tr>
<tr>
<td>userdb</td>
<td>DDname of the user CA IDMS/DB file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Data set name of the user CA IDMS/DB file</td>
</tr>
<tr>
<td>userdd</td>
<td>DDname of the CA IDMS VSAM Transparency file, as it appears in the application</td>
</tr>
<tr>
<td>fmtname</td>
<td>Name of the file management table that defines the escaped file</td>
</tr>
<tr>
<td>ssname</td>
<td>Subschema name</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Buffer size</td>
</tr>
<tr>
<td>devtype</td>
<td>Disk or tape</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>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 CA IDMS Common Facilities Section.</td>
</tr>
</tbody>
</table>
More information:

For more information on these and additional CA IDMS VSAM Transparency parameters, refer to CA IDMS VSAM Transparency parameters (see page 58) 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=ids.dba.loadlib,DISP=SHR
// DD DSN=ids.loadlib,DISP=SHR
//DD DSN=IDMS.EMPLOAD,DISP=SHR
//SYSJRNLD DSN=ids.journal.local, DISP=(NEW, KEEP), unit=devtype
//EMPFILE DD SUBSYS=(ESVS, ‘FMT=EMPFMT’, ‘SUBSCHEMA=EMPSS01’) additional DD statements to run VSAM application
```

Modify the application JCL -- z/VSE

Setting up DLBL Statements

The use of DLBL statements with CA IDMS VSAM Transparency has specific rules that do not correspond to standard z/VSE 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 (z/VSE)**

```
// DLBL userdd,'IDMS.CICS.VSAMT',,VSAM
// DLBL SYSESVS,'sysesvsparms'
// EXEC DFHSIP
```

<table>
<thead>
<tr>
<th>user</th>
<th>Filename of the CA IDMS VSAM Transparency file, as it appears in the application</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDM</th>
<th>Name of a VSAM dataset that is defined in the VSAM catalog. This dummy file should be a KSDS S. dataset, even if the file being escaped was an ESDS file, to prevent VSAM catalog management CICS from doing special processing associated with ESDS datasets. It should contain one or more records to prevent VSAM from opening the file I/O.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSA</td>
<td></td>
</tr>
<tr>
<td>MT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>syse</th>
<th>File ID of the SYSEVS parameters file</th>
</tr>
</thead>
<tbody>
<tr>
<td>svs.</td>
<td></td>
</tr>
<tr>
<td>par</td>
<td></td>
</tr>
<tr>
<td>ms</td>
<td></td>
</tr>
</tbody>
</table>

More information:

For more information on CA IDMS VSAM Transparency parameters, see later in this section. For more information on CICS SYSEVS parameters, see .

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 (z/VSE)**

```
// EXEC PROC=IDMSLBLS
// DLBL idmslib,'idms.lib'
// EXTENT xxxxxx
// LIBDEF PHASE, SEARCH=idmslib.sublib
// DLBL userdd, 'FMT=fmtname', ,VSAM, ,CAT=ESVS
// DLBL ESVS, 'SUBSCHEMA=ssname,MODE=modetype', ,VSAM, ,CAT=ESVS
// DLBL ESVS1, 'RBUFSZ=nnnnn'
// EXEC userpgm Insert SYSIDMS parameters, as required
```

<table>
<thead>
<tr>
<th>IDMSLBLS</th>
<th>Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, SYSIDMS, parameter file and other files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxxxx</td>
<td>Volume serial number of disk unit</td>
</tr>
<tr>
<td>idmslib</td>
<td>Filename of the CA IDMS Library</td>
</tr>
<tr>
<td>idms.lib</td>
<td>File-id of the CA IDMS Library</td>
</tr>
<tr>
<td>userdd</td>
<td>Filename of the CA IDMS VSAM Transparency file, as it appears in the application</td>
</tr>
<tr>
<td>ESVS.FMT.</td>
<td>Name of a VSAM dataset that is defined in the VSAM catalog.</td>
</tr>
<tr>
<td>FMTNAME</td>
<td>You must specify ESVS.FMT. and then replace <em>fmtname</em> with your FMT name.</td>
</tr>
<tr>
<td>fmtname</td>
<td>Name of the file management table that defines the escaped file.</td>
</tr>
<tr>
<td>ssname</td>
<td>Subschema name</td>
</tr>
<tr>
<td>modetype</td>
<td>Optional indicator of BATCH or CICS mode; if not specified, the default is BATCH.</td>
</tr>
<tr>
<td>nnnnn</td>
<td>Buffer size</td>
</tr>
<tr>
<td>userpgm</td>
<td>Application program name</td>
</tr>
<tr>
<td>SYSIDMS</td>
<td>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 CA IDMS Common Facilities Section.</td>
</tr>
</tbody>
</table>

More information:

For more information on these and additional CA IDMS VSAM Transparency parameters, refer to 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:
Additional database file specifications

```
// EXEC userpgm
Insert SYSIDMS parameters, as required
```

More information:

**IDMS** Name of the procedure provided at installation containing the file definitions for CA IDMS dictionaries, databases, SYSIDMS, parameter file and other files.

**nnn** Logical unit assignment for the SYSJRNL in the DMCL

**xxxxx** Volume serial number of disk unit

**idmslib** Filename of the CA IDMS Library

**b** File-id of the CA IDMS 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

**Mode** Optional indicator of BATCH or CICS mode; if not specified, the default is BATCH type

**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 *CA IDMS Common Facilities Section*. 
For more information on these and additional CA IDMS VSAM Transparency parameters, refer to CA IDMS VSAM Transparency parameters (see page 58) 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=IDMSLBLS
// DLBL idmslib,'idms.lib'
// EXTENT ,xxxxxx
// LIBDEF PHASE, SEARCH=idmslib.sublib
// 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 EMPLIST
ddbname=empdb
```

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 z/OS and z/VSE.
- All other parameters must be coded as follows:
  - In z/OS, the parameters can be included in any DD statement, but must appear only once per application.
  - In z/VSE, the parameters must be included in the DLBLfile statement that starts with ESVS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWADISP=nnnn</td>
<td>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.</td>
</tr>
<tr>
<td>DBNAME=database-name</td>
<td>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.</td>
</tr>
<tr>
<td>FMT=fmt-name</td>
<td>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.</td>
</tr>
</tbody>
</table>
**Parameter Description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MSG=LOG</strong></td>
<td>Displays the CA IDMS VSAM Transparency key feedback (runtime) messages during program execution. The options are:</td>
</tr>
<tr>
<td>/CONS</td>
<td>LOG (the default) writes the key feedback messages to the log file.</td>
</tr>
<tr>
<td>/OFF</td>
<td>CONS writes the key feedback messages to the operator console.</td>
</tr>
<tr>
<td></td>
<td>OFF turns off the display of key feedback messages.</td>
</tr>
<tr>
<td></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>FDBK=nnnnnnnnn</td>
</tr>
<tr>
<td></td>
<td><em>Nnnnnnnn</em> is an 8-byte field that contains the key feedback message code.</td>
</tr>
<tr>
<td><strong>MODE=</strong></td>
<td>Indicates whether the CA IDMS VSAM Transparency application program is to run in BATCH or CICS mode. The default is BATCH.</td>
</tr>
<tr>
<td>BATCH</td>
<td></td>
</tr>
<tr>
<td>/CICS</td>
<td></td>
</tr>
<tr>
<td><strong>NODENAME</strong></td>
<td>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. The NODENAME parameter can be from 1 to 8 characters long.</td>
</tr>
<tr>
<td><strong>OCMSGS=</strong></td>
<td>Specifies which messages you want displayed on the console. The options are:</td>
</tr>
<tr>
<td>ALL</td>
<td>ALL indicates both informational and error messages are displayed on the console.</td>
</tr>
<tr>
<td>/ERROR</td>
<td>ERROR indicates that only error messages are displayed on the console.</td>
</tr>
<tr>
<td><strong>RBUFSZ=n</strong></td>
<td>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. The RBUFSZ parameter can be from 1 to 5 digits long and is valid for the central version only. 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.</td>
</tr>
<tr>
<td><strong>STATS=</strong></td>
<td>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:</td>
</tr>
<tr>
<td>ON</td>
<td>ON places the statistics for all CA IDMS VSAM Transparency files together on the system log.</td>
</tr>
<tr>
<td>FILE</td>
<td>FILE places the statistics for each CA IDMS VSAM Transparency file separately on the system log.</td>
</tr>
<tr>
<td><strong>SUBSCHEMA</strong></td>
<td>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 CICS 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. The SUBSCHEMA parameter can be from 1 to 8 characters long.</td>
</tr>
<tr>
<td><strong>TNT=tnt-name</strong></td>
<td>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. The TNT parameter can be from 1 to 8 characters long.</td>
</tr>
</tbody>
</table>
Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE=nnn,WRAP</td>
<td>Requests that a trace of calls made by the CA IDMS VSAM Transparency application program be displayed on the system log. $nnn$ 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: 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.</td>
</tr>
</tbody>
</table>

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.

System Execution

Contents

- Step 1 Run the CA IDMS VSAM Transparency Command Interface (see page 60)
  - CA IDMS VSAM Transparency system commands (see page 61)
  - CA IDMS VSAM Transparency initialization -- z/OS (see page 61)
  - CA IDMS VSAM Transparency initialization -- z/VSE (see page 62)
- Step 2 Bring Up the DC/UCF System (see page 63)

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.

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 z/OS and z/VSE.
CA IDMS VSAM Transparency system commands

Considerations

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

- In z/OS, you enter these commands directly from an operator console.
- In z/VSE, 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** (z/VSE 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. More information: For more information on abnormal termination, see, later in this section.
- **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.

CA IDMS VSAM Transparency initialization -- z/OS

Startup Procedure

For z/OS, the CA IDMS VSAM Transparency command interface startup procedure must reside in SYS1.PROCLIB. For information on the command interface startup procedure, refer to.

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

```
Step 1
s esvs
```
CA IDMS VSAM Transparency initialization -- z/VSE

Startup Procedure

For z/VSE, the CA IDMS VSAM Transparency command interface startup program may be in any private library. CASAUTIL must be run prior to the startup program.

To initiate the CA IDMS VSAM Transparency system in z/VSE, execute the job stream shown below:

**ESVSINIT (z/VSE)**

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idmslib</td>
<td>Filename of the CA IDMS Library</td>
</tr>
<tr>
<td>idms.lib</td>
<td>File-id of the CA IDMS Library</td>
</tr>
<tr>
<td>xxxxxx</td>
<td>Volume serial number of disk unit</td>
</tr>
</tbody>
</table>

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 z/VSE environment is presented in Appendix D, CA IDMS VSAM Transparency Return Codes and Messages.

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.

More information:
For instructions on bringing up a DC/UCF system, refer to CA IDMS System Operations Section.

Application Execution

Contents
- Normal Termination (see page 63)
- Abnormal Termination (see page 64)
  - CA IDMS VSAM Transparency shutdown procedures -- z/OS (see page 64)

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.

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.

**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.

More information:

For more information on backup and recovery procedures can be found in *CA IDMS Database Administration Section*.

**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 section only.

**CA IDMS VSAM Transparency shutdown procedures -- z/OS**

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]
```
CA IDMS VSAM Transparency Architecture

This section provides information on CA IDMS VSAM Transparency's components:
- CA IDMS VSAM Transparency Architecture -- Batch Processing (see page 65)
- CA IDMS VSAM Transparency Architecture -- CICS Processing (see page 66)
- Request Processing Modules (see page 68)
- Control Tables (see page 70)

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 an 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.

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.

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 section under Control Tables (see page 70).

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.

More information:
For more information on the IDMSINTC macro and its parameters, refer to CA IDMS installation section for your operating system

Request Processing Modules

Contents
- The CA IDMS VSAM Transparency Front End (see page 68)
- The CA IDMS VSAM Transparency Back End (see page 69)

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.

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

### 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
Control Tables

Contents
- File Management Table (see page 70)
- Transaction Name Table (see page 71)

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.

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.
More information:

For detailed information on the FMT, refer to Preparing Control Information (see page 28).

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

More information:

For detailed information on the TNT, refer to .

CA IDMS VSAM Transparency Installation

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

- z/OS Installation (see page 71)
- z/VSE Installation (see page 74)

CICS considerations are addressed under the corresponding operating system.

z/OS Installation

Steps to Follow

To install CA IDMS VSAM Transparency under z/OS, 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 Section -- z/OS.
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.

   More information:
   For more information on the ADD PROGRAM statement, refer to *CA IDMS System Generation Section*.

4. **Make additions to the CA IDMS system for variable length records**, as detailed in Appendix C, Variable-Length Record Considerations.

5. **Set up an 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. (z/OS) requires all subsystem startup procedures to be in SYS1.PROCLIB.) The name of this procedure is assumed to be ESVS in this section. 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 (z/OS)**

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

<table>
<thead>
<tr>
<th>idms.apf.loadlib</th>
<th>Data set name of the APF-authorized CA IDMS/DB load library that includes ESVSINIT and ESVSSSSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnnn</td>
<td>Volume serial number for the APF-authorized CA IDMS/DB load library</td>
</tr>
<tr>
<td>unit</td>
<td>Disk type</td>
</tr>
</tbody>
</table>

1. **Define CA IDMS VSAM Transparency as a subsystem in z/OS.** You do this by adding the subsystem name, ESVS, to member IEFSSN00 of SYS1.PARMLIB.
2. **You can define ESVS as a subsystem by executing the program ESVSSBLD with `parm=\(<\text{ESVS}\>\).** 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 z/OS has been done.

3. **Cycle the z/OS system** to implement the definition of the CA IDMS VSAM Transparency subsystem.

   ![Note](Note: It is not necessary to cycle the system if you reinstall CA IDMS VSAM Transparency.)

---

### 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 IDMSINTC at CICS startup and shutdown. Additionally, be sure that PPT 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, edit the CICSOPTS member in CUSTOM.SRCLIB, link it with the IDMSINTC LNK member in CUSTOM.LNKLIB and place the output into your CUSTOM.LOADLIB.

   ![Note](Note: For more information on the CICSOPT macro and its parameters, refer to the CA IDMS System Operations Section.)

#### Important

Be sure the PLT entries are created to execute 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.

**Note:** For more information on adding PPT entries, see the CA IDMS Installation Section for your operating environment. For information on the IDMSINTC macro and its parameters, refer to *CA IDMS System Operations Section*.

## z/VSE Installation

### Contents

- Installing CA IDMS VSAM Transparency in the CICS Environment (see page 75)

### Steps to Follow

To install CA IDMS VSAM Transparency under z/VSEA, 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 Section -- z/VSE.

2. **Install the CA IDMS SVC** as described in CA IDMS installation section for your operating system. If using variable-length CA IDMS VSAM Transparency records, there are additional installation considerations. See Variable-Length Record Considerations (see page 76) 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 (z/VSE)

```bash
// DLBL idmslib,'idms.lib'
// EXTENT ,xxxxxx
// LIBDEF SEARCH=idmslib.sublib,CATALOG=idmslib.sublib
// OPTION CATAL
// PHASE ESVSSPVT,*,SVA
// EXEC ASMA90
// ESVSSPVT SVC=nnn
END
/*
// EXEC LNKEDT,SIZE=500K
&
```

<table>
<thead>
<tr>
<th>idmslib</th>
<th>Filename of the CA IDMS Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.lib</td>
<td>File-id of the CA IDMS Library</td>
</tr>
<tr>
<td>xxxxxx</td>
<td>Volume serial number of disk unit</td>
</tr>
<tr>
<td>nnn</td>
<td>This is the number of the Advantage CA IDMS SVC</td>
</tr>
</tbody>
</table>
1. 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).

2. **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.

   More information:
   For more information on the ADD PROGRAM statement, refer to *CA IDMS System Generation Section*.

3. **Make additions to the CA IDMS system for variable length records**, as detailed in .

---

**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 CAIIJMP 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 CICSOPTS**
If you need to reassemble CICSOPTS to change any installation options, edit the CICSOPTS member, re-assemble your CICSOPTS module and linkedit IDMSINTC. Take the job control to do this from the job control that was generated by CAIIJMP for your initial base tape installation.

⚠️ **Note:** For more information on the CICSOPT macro and its parameters, refer to the CA IDMS System Operations Section

**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.

⚠️ **Note:** For information on adding PPT entries, see the CA IDMS Installation Section for your operating environment.

### Variable-Length Record Considerations

This section contains the procedure you need to follow when 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.
1. 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
```

More information:
For a complete description of the ADD RECORD statement, refer to CA IDMS Database Administration Section.

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.

More information:
For more information on FMT syntax, refer to .

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.

CA IDMS VSAM Transparency User Exits

This section 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.

For more information, see the following topics:
- COBOL User Exit Program (see page 78)
- Assembler User Exit Program (see page 90)

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.

Contents
- Program Requirements (see page 78)
- COBOL User Exit Control Block (see page 80)
- Sample COBOL User Exit (see page 85)

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.

More information:

For more information on coding CA IDMS/DB and DC/UCF programs see CA IDMS Navigational DML Programming Section.

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(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

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: User exit programs should have the language specified as COBOL.

\[\text{Note:} \quad \text{User exit programs should have the language specified as COBOL.}\]
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 cannot. The COBOL code describing the record layout follows:

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

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

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

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Len</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EXB-ID</td>
<td>A</td>
<td>4</td>
<td>Contains the literal 'EXB'.</td>
</tr>
<tr>
<td>4</td>
<td>EXB-USER</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>8</td>
<td>EXB-ADDR-RWA</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>12</td>
<td>EXB-ADDR-STACK</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>16</td>
<td>EXB-ADDR-SSC</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>20</td>
<td>FILLER</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>24</td>
<td>EXB-ADDR-RPL</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>N</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Note: In the Type column, A means alphanumeric and N means numeric (COMP).
<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Len</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>EXB-ADDR-ARGUMENT</td>
<td>N</td>
<td>4</td>
<td>Can not be used by the COBOL exit. Reserved for the Assembler exit.</td>
</tr>
<tr>
<td>36</td>
<td>EXB-MAX-KEY-LENGTH</td>
<td>N</td>
<td>2</td>
<td>Maximum key length.</td>
</tr>
<tr>
<td>38</td>
<td>EXB-REC-BUFFER-LENGTH</td>
<td>N</td>
<td>2</td>
<td>Current length of record in the record buffer.</td>
</tr>
<tr>
<td>40</td>
<td>EXB-VSAM-REC-LENGTH</td>
<td>N</td>
<td>2</td>
<td>Contains the VSAM record length obtained from FMT.</td>
</tr>
<tr>
<td>42</td>
<td>EXB-VSAM-KEY-LENGTH</td>
<td>N</td>
<td>2</td>
<td>Contains the VSAM key length obtained from FMT.</td>
</tr>
<tr>
<td>44</td>
<td>EXB-VSAM-KEY-POSITION</td>
<td>N</td>
<td>2</td>
<td>Contains the VSAM key position obtained from FMT.</td>
</tr>
<tr>
<td>46</td>
<td>EXB-FMT-NAME</td>
<td>A</td>
<td>8</td>
<td>FMT name.</td>
</tr>
<tr>
<td>54</td>
<td>EXB-DD-NAME</td>
<td>A</td>
<td>8</td>
<td>DD name.</td>
</tr>
<tr>
<td>62</td>
<td>EXB-IDMS-REC-NAME</td>
<td>A</td>
<td>16</td>
<td>CA IDMS/DB record name.</td>
</tr>
<tr>
<td>78</td>
<td>EXB-IDMS-SET-NAME</td>
<td>A</td>
<td>16</td>
<td>CA IDMS/DB set name.</td>
</tr>
<tr>
<td>94</td>
<td>EXB-RESERVED</td>
<td>N</td>
<td>2</td>
<td>Reserved field.</td>
</tr>
<tr>
<td>96</td>
<td>EXB-FEEDBACK</td>
<td>A</td>
<td>4</td>
<td>Allows you to set a feedback code (return code) from the exit program. By</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>placing a value here (such as subschema control status) and by moving an 'X'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>' to EXIT-SET-FEEDBACK, CA IDMS VSAM Transparency will display an error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>message on the job log including the feedback code you set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Internal error id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Return code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Component code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error code</td>
</tr>
<tr>
<td>100</td>
<td>EXB-RPL-REQUEST</td>
<td>A</td>
<td>6</td>
<td>Identifies the type of VSAM call issued. Use this field in conjunction with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EXB-EXIT-FLAGS to determine what type of processing will be required. Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>values are:</td>
</tr>
<tr>
<td>100</td>
<td>GET-REQUEST</td>
<td>A</td>
<td>6</td>
<td>'GET '</td>
</tr>
<tr>
<td>100</td>
<td>PUT-REQUEST</td>
<td>A</td>
<td>6</td>
<td>'PUT '</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Len</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------</td>
<td>------</td>
<td>-----</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>100</td>
<td>POINT-REQUEST</td>
<td>A</td>
<td>6</td>
<td>'POINT '</td>
</tr>
<tr>
<td>100</td>
<td>ENDREQ-REQUEST</td>
<td>A</td>
<td>6</td>
<td>'ENDREQ'</td>
</tr>
<tr>
<td>100</td>
<td>ERASE-REQUEST</td>
<td>A</td>
<td>6</td>
<td>'ERASE '</td>
</tr>
<tr>
<td>106</td>
<td>EXB-RPL-OPTION-1</td>
<td>A</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>EXB-RPL-OPTION-1</td>
<td>A</td>
<td>8</td>
<td>Further qualifies the request type:</td>
</tr>
<tr>
<td>106</td>
<td>DIRECT-REQUEST</td>
<td>A</td>
<td>1</td>
<td>Direct request</td>
</tr>
<tr>
<td>107</td>
<td>SEQUENTIAL-REQUEST</td>
<td>A</td>
<td>1</td>
<td>Sequential request</td>
</tr>
<tr>
<td>108</td>
<td>SKIP-REQUEST</td>
<td>A</td>
<td>1</td>
<td>Skip request</td>
</tr>
<tr>
<td>109</td>
<td>KEY-GT-EQ-REQUEST</td>
<td>A</td>
<td>1</td>
<td>Key gt/eq request</td>
</tr>
<tr>
<td>110</td>
<td>GENERIC-REQUEST</td>
<td>A</td>
<td>1</td>
<td>Generic request</td>
</tr>
<tr>
<td>111</td>
<td>FILLER</td>
<td>A</td>
<td>3</td>
<td>Filler</td>
</tr>
<tr>
<td>114</td>
<td>EXB-RPL-OPTION-2</td>
<td>A</td>
<td>8</td>
<td>Further qualifies the request type:</td>
</tr>
<tr>
<td>114</td>
<td>KEYED-ACCESS</td>
<td>A</td>
<td>1</td>
<td>Keyed access</td>
</tr>
<tr>
<td>115</td>
<td>ADDRESS-ACCESS</td>
<td>A</td>
<td>1</td>
<td>Address access</td>
</tr>
<tr>
<td>116</td>
<td>BACKWARD-ACCESS</td>
<td>A</td>
<td>1</td>
<td>Backward access</td>
</tr>
<tr>
<td>117</td>
<td>LAST-REC-ACCESS</td>
<td>A</td>
<td>1</td>
<td>Last record access</td>
</tr>
<tr>
<td>118</td>
<td>UPDATE-ACCESS</td>
<td>A</td>
<td>1</td>
<td>Update access</td>
</tr>
<tr>
<td>119</td>
<td>SET-POSITION</td>
<td>A</td>
<td>1</td>
<td>Set position</td>
</tr>
<tr>
<td>120</td>
<td>FILLER</td>
<td>A</td>
<td>2</td>
<td>Filler</td>
</tr>
<tr>
<td>122</td>
<td>EXB-VSAM-FILE-TYPE</td>
<td>A</td>
<td>8</td>
<td>Identifies the VSAM file type:</td>
</tr>
<tr>
<td>122</td>
<td>KSDS</td>
<td>A</td>
<td>1</td>
<td>KSDS</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Len</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------</td>
<td>------</td>
<td>-----</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>123</td>
<td>PATH</td>
<td>A</td>
<td>1</td>
<td>PATH</td>
</tr>
<tr>
<td>124</td>
<td>RRDS</td>
<td>A</td>
<td>1</td>
<td>RRDS</td>
</tr>
<tr>
<td>125</td>
<td>ESDS</td>
<td>A</td>
<td>1</td>
<td>ESDS</td>
</tr>
<tr>
<td>126</td>
<td>BASE-CLUS-ESDS</td>
<td>A</td>
<td>1</td>
<td>Base cluster ESDS</td>
</tr>
<tr>
<td>127</td>
<td>SET-DEFINED</td>
<td>A</td>
<td>1</td>
<td>Set defined</td>
</tr>
<tr>
<td>128</td>
<td>NATIVE-VSAM</td>
<td>A</td>
<td>1</td>
<td>Native VSAM</td>
</tr>
<tr>
<td>129</td>
<td>FILLER</td>
<td>A</td>
<td>1</td>
<td>Filler</td>
</tr>
<tr>
<td>130</td>
<td>EXB-EXIT-FLAGS</td>
<td>A</td>
<td>8</td>
<td>Exit flags that control CA IDMS VSAM Transparency processing</td>
</tr>
<tr>
<td>130</td>
<td>EXB-EXIT-TYPE</td>
<td>A</td>
<td>1</td>
<td>Indicates the exit type:</td>
</tr>
<tr>
<td>130</td>
<td>BEFORE-EXIT</td>
<td>A</td>
<td>1</td>
<td>Before = 'B'</td>
</tr>
<tr>
<td>130</td>
<td>AFTER-EXIT</td>
<td>A</td>
<td>1</td>
<td>After = 'A'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The next 5 bytes describe exit processing. Put an X in the byte to turn the character switch on. The default is blank.</td>
</tr>
<tr>
<td>131</td>
<td>SKIP-TO-AFTER</td>
<td>A</td>
<td>1</td>
<td>Tells CA IDMS VSAM Transparency to invoke the after exit immediately after the before exit and bypass CA IDMS VSAM Transparency processing.</td>
</tr>
<tr>
<td>132</td>
<td>RETURN-IMMED</td>
<td>A</td>
<td>1</td>
<td>Tells CA IDMS VSAM Transparency to return immediately to the user and bypass CA IDMS VSAM Transparency processing.</td>
</tr>
<tr>
<td>133</td>
<td>EXIT-SET-REC-LENGTH</td>
<td>A</td>
<td>1</td>
<td>Tells CA IDMS VSAM Transparency that the exit modified the length of the record in the buffer.</td>
</tr>
<tr>
<td>134</td>
<td>EXIT-SET-ARGUMENT</td>
<td>A</td>
<td>1</td>
<td>Tells CA IDMS VSAM Transparency that the exit set the argument.</td>
</tr>
<tr>
<td>135</td>
<td>EXIT-SET-FEEDBACK</td>
<td>A</td>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>136</td>
<td>FILLER</td>
<td>A</td>
<td>2</td>
<td>Filler</td>
</tr>
<tr>
<td>138</td>
<td>FILLER</td>
<td>A</td>
<td>8</td>
<td>Filler</td>
</tr>
<tr>
<td>146</td>
<td>FILLER</td>
<td>A</td>
<td>2</td>
<td>Filler</td>
</tr>
<tr>
<td>148</td>
<td>EXP-SAVE-AREA</td>
<td>A</td>
<td>72</td>
<td>Filler</td>
</tr>
<tr>
<td>218</td>
<td>FILLER</td>
<td>A</td>
<td>12</td>
<td>Filler</td>
</tr>
</tbody>
</table>
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 media 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)


CA IDMS Structure

The user exit must navigate the database and build the record that the VSAM application program is expecting.
Sample COBOL User Exit (2)

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**

```cobol
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'.

CA IDMS - 19.0
16-Jan-2018 88/108
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.
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.

- Program Requirements (see page 90)
- Assembler User Exit Control Block (see page 92)
- Assembler User Exit Template (see page 95)
- Sample Assembler User Exit (see page 96)
- Macros (see page 102)

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 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 '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 Section for 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).

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.

<table>
<thead>
<tr>
<th>Hex</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>EXBID</td>
<td>C</td>
<td>4</td>
<td>Contains the literal 'EXB'</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>EXBU</td>
<td>F</td>
<td>4</td>
<td>Can be used by the exit</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>EXBA</td>
<td>F</td>
<td>4</td>
<td>Exit must not modify this field</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>EXBS</td>
<td>F</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>EXBS</td>
<td>F</td>
<td>4</td>
<td>Address of the Subschema Control</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>FILLER</td>
<td>F</td>
<td>4</td>
<td>Filler</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>EXBR</td>
<td>F</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td></td>
<td>EXBA</td>
<td>F</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>EXBB</td>
<td>F</td>
<td>4</td>
<td>Address of the record in the buffer</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>EXBM</td>
<td>H</td>
<td>2</td>
<td>Maximum key length.</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>EXBB</td>
<td>H</td>
<td>2</td>
<td>Current length of record in the record buffer</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>EXBR</td>
<td>H</td>
<td>2</td>
<td>Contains the VSAM record length obtained from FMT</td>
</tr>
<tr>
<td>2A</td>
<td></td>
<td>EXBK</td>
<td>H</td>
<td>2</td>
<td>Contains the VSAM key length obtained from FMT</td>
</tr>
<tr>
<td>2C</td>
<td></td>
<td>EXBK</td>
<td>H</td>
<td>2</td>
<td>Contains the VSAM key position obtained from FMT</td>
</tr>
<tr>
<td>2E</td>
<td></td>
<td>EXBF</td>
<td>C</td>
<td>8</td>
<td>FMT name</td>
</tr>
<tr>
<td>Hex</td>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>----------</td>
<td>------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>36</td>
<td>EXBD</td>
<td>DNM</td>
<td>C</td>
<td>8</td>
<td>DD name</td>
</tr>
<tr>
<td>3E</td>
<td>EXBR</td>
<td>ECN</td>
<td>M</td>
<td>16</td>
<td>CA IDMS/DB record name</td>
</tr>
<tr>
<td>4E</td>
<td>EXBS</td>
<td>ETN</td>
<td>M</td>
<td>16</td>
<td>CA IDMS/DB set name</td>
</tr>
<tr>
<td>5E</td>
<td>EXBR</td>
<td>ESV</td>
<td>H</td>
<td>2</td>
<td>Reserved field</td>
</tr>
<tr>
<td>60</td>
<td>EXBF</td>
<td>DBK</td>
<td>C</td>
<td>4</td>
<td>Allows you to set a feedback code (return code) from the exit program. By</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>placing a value here (such as subschema control status) and by moving an 'X'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to EXIT-SET-FEEDBACK, CA IDMS VSAM Transparency will display an error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>message on the job log including the feedback code you set.</td>
</tr>
<tr>
<td>60</td>
<td>EXBE</td>
<td>RID</td>
<td>C</td>
<td>1</td>
<td>Internal error id</td>
</tr>
<tr>
<td>61</td>
<td>EXBR</td>
<td>TCD</td>
<td>C</td>
<td>1</td>
<td>Return code</td>
</tr>
<tr>
<td>62</td>
<td>EXBC</td>
<td>OMP</td>
<td>C</td>
<td>1</td>
<td>Component code</td>
</tr>
<tr>
<td>63</td>
<td>EXBE</td>
<td>RCD</td>
<td>C</td>
<td>1</td>
<td>Error code</td>
</tr>
<tr>
<td>64</td>
<td>EXBR</td>
<td>TYPE</td>
<td>C</td>
<td>6</td>
<td>Identifies the type of VSAM call issued. Use this field in conjunction with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EXB-EXIT-FLAGS to determine what type of processing will be required. Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>values are: 'GET', 'PUT', 'POINT', 'ENDEUR', 'ERASE'</td>
</tr>
<tr>
<td>6A</td>
<td>EXBO</td>
<td>PT1</td>
<td>C</td>
<td>8</td>
<td>Further qualifies the request type:</td>
</tr>
<tr>
<td>6A</td>
<td>EXBDI</td>
<td>R</td>
<td>C</td>
<td>1</td>
<td>Direct request</td>
</tr>
<tr>
<td>6B</td>
<td>EXBS</td>
<td>EQ</td>
<td>C</td>
<td>1</td>
<td>Sequential request</td>
</tr>
<tr>
<td>6C</td>
<td>EXBS</td>
<td>KP</td>
<td>C</td>
<td>1</td>
<td>Skip request</td>
</tr>
<tr>
<td>6D</td>
<td>EXBK</td>
<td>GE</td>
<td>C</td>
<td>1</td>
<td>Key gt/eq request</td>
</tr>
<tr>
<td>6E</td>
<td>EXBG</td>
<td>EN</td>
<td>C</td>
<td>1</td>
<td>Generic request</td>
</tr>
<tr>
<td>6F</td>
<td>FILLE</td>
<td>R</td>
<td>C</td>
<td>3</td>
<td>Filler</td>
</tr>
<tr>
<td>Hex</td>
<td>Name</td>
<td>Type</td>
<td>Length</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>EXBO</td>
<td>C</td>
<td>8</td>
<td>Further qualifies the request type:</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>EXBK</td>
<td>C</td>
<td>1</td>
<td>Keyed access</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>EXBA</td>
<td>C</td>
<td>1</td>
<td>Address access</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>EXBB</td>
<td>C</td>
<td>1</td>
<td>Backward access</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>EXBL</td>
<td>C</td>
<td>1</td>
<td>Last record access</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>EXBU</td>
<td>C</td>
<td>1</td>
<td>Update access</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>EXBN</td>
<td>C</td>
<td>1</td>
<td>Set position</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>FILLE</td>
<td>C</td>
<td>2</td>
<td>Filler</td>
<td></td>
</tr>
<tr>
<td>7A</td>
<td>EXBF</td>
<td>C</td>
<td>8</td>
<td>Identifies the VSAM file type:</td>
<td></td>
</tr>
<tr>
<td>7A</td>
<td>EXBK</td>
<td>C</td>
<td>1</td>
<td>KSDS</td>
<td></td>
</tr>
<tr>
<td>7B</td>
<td>EXBP</td>
<td>C</td>
<td>1</td>
<td>PATH</td>
<td></td>
</tr>
<tr>
<td>7C</td>
<td>EXBR</td>
<td>C</td>
<td>1</td>
<td>RRDS</td>
<td></td>
</tr>
<tr>
<td>7D</td>
<td>EXBE</td>
<td>C</td>
<td>1</td>
<td>ESDS</td>
<td></td>
</tr>
<tr>
<td>7E</td>
<td>EXBB</td>
<td>C</td>
<td>1</td>
<td>Base cluster is ESDS</td>
<td></td>
</tr>
<tr>
<td>7F</td>
<td>EXBS</td>
<td>C</td>
<td>1</td>
<td>Set is defined</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>EXBN</td>
<td>C</td>
<td>1</td>
<td>Data is in native VSAM</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>FILLE</td>
<td>C</td>
<td>1</td>
<td>Filler</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>EXBX</td>
<td>C</td>
<td>8</td>
<td>Exit flags that describe exit processing:</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>EXBE</td>
<td>C</td>
<td>1</td>
<td>The first byte identifies the exit type:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Before = B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>After = A</td>
<td></td>
</tr>
</tbody>
</table>

The next 5 bytes describe exit processing. Put an X in the byte to turn the character switch on. The default is blank.
<table>
<thead>
<tr>
<th>Hex</th>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXBS</td>
<td>KPA</td>
<td>Skip to after tells CA IDMS VSAM Transparency to invoke the after exit immediately after the before exit and bypass its processing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>EXBX</td>
<td>RTN</td>
<td>C</td>
<td>1</td>
<td>Return immediately tells CA IDMS VSAM Transparency to return immediately to the user and bypass its processing.</td>
</tr>
<tr>
<td>85</td>
<td>EXBS</td>
<td>ETLN</td>
<td>C</td>
<td>1</td>
<td>Exit set record length tells CA IDMS VSAM Transparency the exit modified the length of the record in the buffer.</td>
</tr>
<tr>
<td>86</td>
<td>EXBS</td>
<td>ETAR</td>
<td>C</td>
<td>1</td>
<td>Exit set argument tells CA IDMS VSAM Transparency the exit set argument.</td>
</tr>
<tr>
<td>87</td>
<td>EXBS</td>
<td>ETFB</td>
<td>C</td>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>88</td>
<td>FILLE</td>
<td>C</td>
<td>2</td>
<td>Filler</td>
<td></td>
</tr>
<tr>
<td>8A</td>
<td>FILLE</td>
<td>C</td>
<td>8</td>
<td>Filler</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>EXBS</td>
<td>AVE</td>
<td>C</td>
<td>72</td>
<td>Exit save area</td>
</tr>
</tbody>
</table>

**Assembler User Exit Template**

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

```assembly
EXITNAME CSECT
    #MOPT ENV=USER
    @MODE MODE=BATCH,QUOTES=YES,DEBUG=YES
* DEFINE ENVIRONMENT
    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
```
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 . The source code for this program is provided on the installation media with CA IDMS VSAM Transparency (member name ESVSXIITA).

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

VSAM Structure (Variable Length KSDS Record)

Sample COBOL User Exit

CA IDMS Structure

The user exit must navigate the database and build the record that the VSAM application program is expecting.
Sample COBOL User Exit (2)

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**

```assembly
*** THIS EXIT IS INVOKED USING STANDARD z/OS 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 EXITDMS ***
*** 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 ***
```

---

16-Jan-2018 98/108
USING RECORD,R8 POLICY RECORD ADDRESSIBILITY
USING CSA,R10 COMMON SYSTEM AREA ADDRESSIBILITY

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

*** ACQUIRE STORAGE FOR EXIT TO USE
***
GETSTG #GETSTG TYPE=(USER,SHORT),PLIST=*,LEN=50,INIT=0,
      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

* BEFORE EXIT - PUT REQUEST
* MOVE ZERO TO CONTROL FIELD
BEFORPUT MVC CNTRLFLD,=H'00' MOVE ZERO TO OCCURS FIELD
   B RTN RETURN

* 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

* BEFORE EXIT - ERASE REQUEST
* GET POLICY RECORD AND ERASE ALL COVERAGE RECORDS WITHIN
* POLICY-COVG SET.

BEFOREA 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

*** 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

*** 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=COVREC,IOAREA=(R4) BIND COVERAGE RECORD TO R4
   CLC SSCSTAT,=CL4'0000' STATUS ZERO
   BNE RTNERR NO, RETURN ERROR

*** OBTAIN COVERAGE RECORDS
OBTCOVG @OBTAIN NEXT,SET=SETNAME,REC=COVREC
   CLC SSCSTAT,=CL4'0000' STATUS ZERO??
   BNE RTNERR NO, RETURN ERROR
*** ERASE COVERAGE RECORD
@ERASE ALL,REC=COVGREC
* THIS POLICY
CLC SSCSTAT,=CL4'0000'
BNE RTNERR
BCT R6,OBTCOVG
SPACE
B RTN
RETURN
EJECT
* CHECK TO SEE IF THIS IS AN AFTER EXIT WITH A GET REQUEST
CHKAFTER CLI EXBXEXIT,C'A'
BNE RTN
CLC EXBRTYPE,=CL6'GET'
BNE CHKPUT
CLC SSCSTAT,=CL4'0000'
BNE RTN
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'
BNE TYP1041
LA R6,4
B SETUP
TYP1041 CLC POLTYP,=CL4'1041'
BNE NOTREC
LA R6,3
SPACE
SETUP MVC OCCURS,=CL40' ' CLEAR OUT AREA TO STORE COVERAGE
LA R5,27(0,R8) R5 START OF COVERAGE RECORDS
**** BIND COVERAGE RECORD
@BIND REC=COVGREC,IOAREA=(R4) BIND COVERAGE RECORD TO R4
CLC SSCSTAT,=CL4'0000'
BNE RTNERR
SPACE
*** OBTAIN COVERAGE RECORD
GETREC @OBTAIN NEXT,SET=SETNAME
* WITHIN POLICY-COVG SET
CLC SSCSTAT,=CL4'0000'
BNE RTNERR
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 CNTLFLD,=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'
BNE RTN
CLC EXBRTYPE,=CL6'PUT'
BNE CHKPUT
CLC SSCSTAT,=CL4'0000'
BNE RTN
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'
BNE TYPE1041
LA R6,4
B SETUP
TYPE1041 CLC POLTYP,=CL4'1041'
BNE RTN
LA R6,3
SETUPS
LA R5,27(0,R8) R5 START OF COVERAGE RECORDS
*
*** BIND COVERAGE RECORD
@BIND REC=COVGREC,IOAREA=(R4) BIND COVERAGE RECORD TO R4
CLC SSCSTAT,=CL4'0000' STATUS ZERO
BNE RTNERR NO, RETURN ERROR

*** 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
END OF EXECUTABLE CODE

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 POLICY RECORD LAYOUT
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 COVERAGE RECORD LAYOUT
DATA DS CL10
EJECT COPY #EVEXBDS
EJECT COPY #SCDS
EJECT PRINT OFF
PRINT NGEN
COPY #CSADS
PRINT GEN
END ESVSXITA
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)
```

**#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
```

**Note:** The #MOPT macro also equates register 0 through register 15 to R0 through R15.
TSO File Allocation with CA IDMS VSAM Transparency

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

Purpose

Under TSO, each CA IDMS VSAM Transparency file must be directed to the subsystem. The ESVSALLO command allocates an 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)
Subsys (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(nn.nn)**
  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. *Nnnn* 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(nnun,WRAP/LOG)**  
  Requests that a trace of calls made by the CA IDMS VSAM Transparency program be displayed on the system log.  
  \( \text{nnun} \) 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.

More information:

For more information on these parameters see Runtime Operations (see page 43).

**Example**

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

```
ESVSALL0 FILE(TESTFILE) FMT(TESTFMT) SSC(TESTSSC)
ESVSALL0 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.

**z/VSE CICS SYSESVS Parameters**

This section provides information on z/VSE CICS SYSESVS parameters.

**Purpose**

At z/VSE 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 z/VSE 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:

<table>
<thead>
<tr>
<th>Cols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-72</td>
<td>Keyword operands and keyword values</td>
</tr>
<tr>
<td>73-80</td>
<td>Reserved for record sequence numbers</td>
</tr>
</tbody>
</table>

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 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 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 effect 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=xxxxxxxx

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

FMT=xxxxxxxx

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 xxxxxxxx is the table name. This parameter is required and must follow the associated FILENAME parameter immediately.

**SYSESVS Parameter File**