CA IDMS - 19.0
Using TOTAL 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

CA IDMS TOTAL Transparency as a Conversion Tool ................................. 8

CA IDMS TOTAL Transparency as a Runtime Tool ...................................... 8

Components of the CA IDMS TOTAL Transparency ..................................... 9
The Generator ETOTMAIN ........................................................................ 9
The Loader ETOTLOAD ......................................................................... 11
The Runtime Control Table ETOTTBL ...................................................... 11
The Runtime Interface ........................................................................... 11

CA IDMS TOTAL Transparency ................................................................ 12
Total Relationships .................................................................................. 12
CA IDMS/DB Relationships ...................................................................... 12
Conversion of Total Data Structures to CA IDMS/DB ................................. 12
  Total Files Relate to CA IDMS/DB Record Types .................................. 13
  Total Data Items Relate to CA IDMS/DB Elements ................................. 13
  Total Master File Control Keys Relate to CA IDMS/DB CALC Keys .......... 13
  Total Linkpath Fields Relate to CA IDMS/DB Pointers ........................... 13
  Total Linkpaths Relate to CA IDMS/DB Sets .......................................... 13
Implementing the Transparency ............................................................... 14
  Data Description Phase ....................................................................... 14
  Database Conversion Phase .................................................................. 15
  Runtime Operations Phase ................................................................... 16
Operating Environments Supported ......................................................... 17
Support for Total .................................................................................... 17
  Total Features Supported ................................................................... 18
  Total Features Not Supported .............................................................. 19

Phase: Data Description ......................................................................... 20
Step 1 -- Select a Total Database ............................................................ 20
Step 2 -- Prepare Total DDL Statements .................................................. 20
Step 3 -- Prepare the Physical CA IDMS/DB Database .............................................. 21
Step 4 -- Prepare the Logical CA IDMS/DB Database .................................................. 21
Steps of the Data Description Phase .............................................................. 21
Generator Control Statements ................................................................. 22
  Summary of Statements ................................................................. 22
  Limits on Control Statements .......................................................... 23
SYNONYMS Statement ...................................................................... 23
AREA Statement ........................................................................... 25
SCHEMA Statement ....................................................................... 26
SUBSCHEMA Statement .................................................................. 28
USAGE-MODE Statement ............................................................... 29
LOAD Statement ............................................................................ 32

Database Conversion Phase ...................................................................... 34
Step 1 Transparency Generator (ETOTMAIN) .............................................. 35
Step 2 Schema Preparation .................................................................. 41
Step 3 Physical Database Preparation ................................................... 42
Step 4 Subschema Preparation ............................................................. 43
Step 5 CA IDMS/DB Database Initialization ........................................... 45
Step 6 Total Database Unload Utility ....................................................... 45
Step 7 Transparency Loader (ETOTLOAD) ................................................ 45

Runtime Operations Phase ...................................................................... 46

Generator Messages ........................................................................... 48
  Error Messages 3 ......................................................................... 49
  Warning Messages 3 ...................................................................... 51
  Fatal Messages ............................................................................ 51

Loader Messages ................................................................................ 51

Increasing Generator Input Limits ......................................................... 51
  Limits on Total DDL Statements ..................................................... 52
  Limits on Transparency Generator Control Statements .................... 52
Using TOTAL Transparency

z/OS Job Control Language ................................................................. 52
Generate (ETOTMAIN) ........................................................................... 52
Assembling and Link Editing ETOTTBL .............................................. 54
Compiling and Link Editing ETOTLOAD ........................................... 54
Loading the Database with ETOTLOAD ............................................. 55
Assembling the CICS Transparency Interface (ETOTCINT) .................. 57
Creating the CICS CA IDMS/DB Interface (IDMSINTC) ..................... 57
Batch Application Program Link Edit .............................................. 58
CICS Application Program Link Edit .............................................. 58
Runtime JCL ..................................................................................... 59

z/VSE Job Control Language .............................................................. 60
Generate ETOTMAIN ........................................................................... 60
Assembling and Link Editing ETOTTBL 1 ......................................... 61
Compile and Link Edit ETOTLOAD .................................................. 61
Loading Database with ETOTLOAD ................................................ 62
Assembling the CICS/Transparency Interface (ETOTCINT) .................. 63
Creating the CICS CA IDMS/DB Interface (IDMSINTC) 1 ................... 64
Batch Application Program Link Edit 1 .......................................... 65
CICS Application Program Link Edit 1 .......................................... 65
JCL Runtime .................................................................................... 65
Using TOTAL Transparency

The CA IDMS TOTAL Transparency is a product that helps users of the Total Database Management System convert to a CA IDMS/DB database environment. Through the transparency, Total users can:
CA IDMS TOTAL Transparency as a Conversion Tool

To convert a Total database to a CA IDMS/DB database, the transparency does the following:

- Translates Total data definition language (DDL) into the logical DDL needed for a CA IDMS/DB database
- Loads the unloaded Total database to a CA IDMS/DB database

CA IDMS TOTAL Transparency as a Runtime Tool

To enable Total application programs to run against the CA IDMS/DB database, the transparency provides a runtime interface. This interface replaces all Total software and performs the following functions:

- Simulates the Total database processing environment to satisfy database requests issued by Total applications.
- Converts Total data manipulation language (DML) calls into CA IDMS DML calls.
- Represents CA IDMS/DB data structures as Total data structures.
- Accepts Total database requests from both batch and CICS programs. CICS programs must issue command level requests.

For more information, see the following topics:
- Components of the CA IDMS TOTAL Transparency (see page 9)
- CA IDMS TOTAL Transparency (see page 12)
- Phase: Data Description (see page 20)
- Database Conversion Phase (see page 34)
- Runtime Operations Phase (see page 46)
- Generator Messages (see page 48)
- Loader Messages (see page 51)
- Increasing Generator Input Limits (see page 51)
- z/OS Job Control Language (see page 52)
- z/VSE Job Control Language (see page 60)
Components of the CA IDMS TOTAL Transparency

The major components of the CA IDMS TOTAL Transparency are as follows:

- The Generator ETOTMAIN (see page 9)
- The Loader ETOTLOAD (see page 11)
- The Runtime Control Table ETOTTBL (see page 11)
- The Runtime Interface (see page 11)

The Generator ETOTMAIN

The generator (ETOTMAIN) is a transparency program that does the following:

- Translates Total DDL into CA IDMS/DB DDL (schema and subschema descriptions)
- Generates a customized load module (ETOTLOAD)
- Generates a customized table (ETOTTBL) that defines the database to the runtime interface

The following table describes input to the generator.

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total DDL</td>
<td>Defines the entire Total database being converted</td>
</tr>
<tr>
<td>Control statements that you prepare</td>
<td>Supply information on the CA IDMS/DB database being created</td>
</tr>
<tr>
<td>Loader (ETOTLOAD) skeleton containing source COBOL statements</td>
<td>Reads unloaded Total data and loads the CA IDMS/DB database</td>
</tr>
<tr>
<td>Runtime control table (ETOTTBL) skeleton</td>
<td>Creates source Assembler statements that provide interface information on the Total database referenced by Total application programs</td>
</tr>
</tbody>
</table>

Equivalent Total and CA IDMS/DB Data Structures

The following diagrams are of a sample Total database and the corresponding CA IDMS/DB database. Note that the CUSTOMER and CUSTOMER-ORDER CA IDMS/DB records correspond to the Total master file CUST and the CUOR variable file. Synonyms are provided to take advantage of the longer names permitted under CA IDMS/DB.
TOTAL
database structure

CA IDMS/DB
database structure

006937e990db4dfe913d51a9046236c3
The Loader ETOTLOAD

The loader (ETOTLOAD) is a customized COBOL source module output by the generator. The loader contains code to read data unloaded from the Total database and to load the corresponding CA IDMS/DB database.

The Runtime Control Table ETOTTBL

The runtime control table (ETOTTBL) is a customized assembler source module output by the generator. Using Total DDL and user-supplied control statements, the generator copies arrays into the WORKING-STORAGE SECTION of ETOTTBL. These arrays describe Total files and their equivalent CA IDMS/DB record types and set relationships.

At runtime, ETOTTBL replaces the Total DBMOD and provides the runtime interface with the control information necessary to:

- Construct the Total records and relationships expected by the Total application program
- Reconstruct CA IDMS/DB records to be stored in the CA IDMS/DB database

The Runtime Interface

The runtime interface translates Total database calls and passes the translated calls to CA IDMS/DB.

<table>
<thead>
<tr>
<th>Operation</th>
<th>How it works</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translate Total</td>
<td>The object module ETOTTRAN translates calls. ETOTTRAN can be link edited to the Total application. If it is not link edited, it will be loaded at runtime. You might want to link ETOTTRAN with batch programs, but dynamically load ETOTTRAN when you use online CICS applications.</td>
</tr>
<tr>
<td>Pass translated</td>
<td><strong>Total batch applications</strong></td>
</tr>
<tr>
<td>calls to CA IDMS/DB</td>
<td>Total batch applications use the object module ETOTBINT. ETOTBINT is link edited with the Total application and CA IDMS/DB to access the CA IDMS/DB database. The runtime control table is loaded by ETOTBINT at runtime.</td>
</tr>
<tr>
<td></td>
<td><strong>Total CICS applications</strong></td>
</tr>
<tr>
<td></td>
<td>Total CICS applications use the macro ETOTCINT to pass calls to program IDMSINTC. IDMSINTC, in turn, passes the calls to CA IDMS/DB. ETOTCINT must be assembled before runtime. The assembled ETOTCINT is link edited with the Total application. ETOTTRAN is link edited as part of the IDMSINTC load module.</td>
</tr>
</tbody>
</table>

The runtime interface uses the generated subschema to access the CA IDMS/DB database. This subschema provides access to all elements, record types, set types, and areas defined in the generated schema.

The Mixed Page Group Binds Allowed feature may not be used with CA IDMS TOTAL Transparency.
CA IDMS TOTAL Transparency

Total Relationships

A Total file definition includes not only descriptions of user data but also descriptions of any one-to-many relationships that exist with other files. This is reflected in the presence of linkpath fields and the designation of files as either master files or variable files.

For more information, see Conversion of Total Data Structures to CA IDMS/DB (see page 12)

Linkpath Fields are Defined

To associate a master file with a variable file, linkpath fields are defined. A linkpath field in a master record points to the first and last of a chain of related records in a variable file. A linkpath field in a variable record points to the next and prior variable record in the chain. Total maintains the one-to-many relationship as part of the data in the record.

CA IDMS/DB Relationships

While the relationships among CA IDMS/DB record types are similar in concept to those among Total files, these relationships are defined and maintained differently. Under CA IDMS/DB, a one-to-many relationship between two or more record types is expressed by a set structure. A set structure is defined and maintained apart from the record definitions.

Set is Defined for Each Relationship

For each relationship between two or more CA IDMS/DB record types, a set is defined. One record type is designated as the owner of the set and the other record type(s) is designated as the member. A given set occurrence consists of one occurrence of the owner record type and any number of occurrences of the member record type(s).

Owner and Members are Linked by Pointers

The owner and members of a set are linked by the pointers maintained in the record prefixes. The record prefix always contains a pointer linking the record to the next record in the set, and can contain pointers linking the record to the prior, last, and/or owner record in the set.

Conversion of Total Data Structures to CA IDMS/DB

To convert a Total database to CA IDMS/DB, the transparency converts Total data structures into CA IDMS/DB data structures. This section provides information on the CA IDMS/DB equivalents of Total data structures.
Total Files Relate to CA IDMS/DB Record Types

A CA IDMS/DB record type represents a collection of similar record occurrences. Record types are defined to CA IDMS/DB by schema record descriptions.

Total Data Items Relate to CA IDMS/DB Elements

CA IDMS/DB elements are the individual units of user data that make up a record. Elements and their characteristics are defined in schema record descriptions.

Total Master File Control Keys Relate to CA IDMS/DB CALC Keys

A CA IDMS/DB CALC key is an element designated as the value within a record used to determine the location of the record in a database area.

The transparency assigns a location mode of CALC to all records derived from Total master files. The transparency assigns a location mode of VIA to all records derived from Total variable files, with the exception of records in coded variable files for which the primary linkpath is not included in the base portion of the file. Records included in these coded variable files are assigned a location mode of DIRECT.

Total Linkpath Fields Relate to CA IDMS/DB Pointers

A CA IDMS/DB pointer links the owner and member records of a CA IDMS/DB set. Note that CA IDMS/DB pointers, unlike Total linkpath fields, are maintained in a record's prefix rather than in the record itself. The transparency uses Total linkpath field information to establish the necessary pointers in the record prefix for each record. The transparency does not retain Total linkpath fields in inverted records.

Total Linkpaths Relate to CA IDMS/DB Sets

Identifies Existing Relationships
When converting a Total database to a CA IDMS/DB database, the transparency uses Total linkpath information to identify each relationship that exists among given Total files. For each such relationship found in Total DDL, the transparency defines a set in the generator-produced CA IDMS/DB schema and subschema.

**Converting to a Set**

When converting a linkpath to a set, the transparency does the following:

- Designates the Total master file as the owner record type
- Designates the associated Total variable file as the member record type
- Establishes appropriate pointers from the linkpath fields used to relate a master file to a variable file and to chain records in a variable file

**PRIOR Set Order is Assigned**

The transparency assigns a set order of PRIOR to each CA IDMS/DB set established. It assigns a set membership option of MANDATORY AUTOMATIC for the primary linkpath and a set membership option of MANDATORY MANUAL for all other CA IDMS/DB member record types that it describes in the schema.

⚠️ **Note:** For more information on the CA IDMS/DB set order and set membership options, see the CA IDMS Database Administration Section.

Records in a Total variable file are stored via the CA IDMS/DB set established to correspond to the primary linkpath for that file.

**Implementing the Transparency**

This section describes the following transparency phases:

- Data Description Phase (see page 14)
- Database Conversion Phase (see page 15)
- Runtime Operations Phase (see page 16)

**Data Description Phase**

In the data description phase, you do the following:

1. Select a Total database for conversion
2. Check that you have the appropriate Total DDL to describe the database selected
3. Prepare control statements; these control statements describe the logical CA IDMS/DB environment that results from database conversion

4. Prepare CA IDMS DDL statements that describe the physical CA IDMS/DB environment

For more information on the data description phase, see Phase: Data Description (see page 20).

Database Conversion Phase

Database Conversion Programs

You input the Total DDL and control statements in the database conversion phase.

In the database conversion phase, you convert your Total database to a CA IDMS/DB database using the programs listed in the following table.

<table>
<thead>
<tr>
<th>Program</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator (ETOTMAIN)</td>
<td>User-prepared control statements</td>
<td>Source DDL for the schema and subschema</td>
</tr>
<tr>
<td></td>
<td>Total DDL</td>
<td>Customized load module</td>
</tr>
<tr>
<td></td>
<td>Skeletons for the loader (ETOTLOAD)</td>
<td>Customized runtime control table module</td>
</tr>
<tr>
<td></td>
<td>Runtime control table (ETOTTBL) modules</td>
<td></td>
</tr>
<tr>
<td>CA IDMS/DB schema compiler (IDMSCHEM)</td>
<td>Schema DDL produced by the transparency generator</td>
<td>Compiled schema descriptions entered in the dictionary (IDD)</td>
</tr>
<tr>
<td>CA IDMS Command Facility (IDMSBCF)</td>
<td>Physical DDL statements that define segments, a database name table,</td>
<td>Compiled DMCL descriptions entered in the catalog (DDLCAT) area</td>
</tr>
<tr>
<td></td>
<td>and the DMCL</td>
<td>of the system dictionary</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> For more information on physical DDL syntax, see the CA</td>
<td>DMCL load module</td>
</tr>
<tr>
<td></td>
<td><strong>IDMS Database Administration Section.</strong> For more information on</td>
<td>Database name table load module</td>
</tr>
<tr>
<td></td>
<td>using IDMSBCF to submit physical DDL statements, see the CA **</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>IDMS Common Facilities Section.</strong></td>
<td></td>
</tr>
<tr>
<td>CA IDMS/DB subschema compiler (IDMSUBSC)</td>
<td>Subschema DDL produced by the generator</td>
<td>Compiled subschema descriptions entered in the dictionary (IDD)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subschema load module</td>
</tr>
<tr>
<td>CA IDMS Command Facility (IDMSBCF)</td>
<td>FORMAT utility statement; use the names of the files included in the</td>
<td>Formatted CA IDMS/DB database</td>
</tr>
<tr>
<td></td>
<td>DMCL load module</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> For more information on the FORMAT syntax, see the CA **</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>IDMS Utilities Section.</strong> For more information on using IDMSBCF to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>submit FORMAT, see the CA <strong>IDMS Common Facilities Section.</strong></td>
<td></td>
</tr>
</tbody>
</table>

Parameters and data in the Total database Unloaded data
When you finish the conversion phase, the CA IDMS/DB database is available for processing. You are ready for the final transparency phase, runtime operations.

**Illustration of the Database Conversion Phase**

The following figure illustrates the database conversion phase.

For more information on the database conversion phase, see Database Conversion Phase (see page 34).

**Runtime Operations Phase**

In the runtime operations phase, you relink your Total and CICS applications to run against the new CA IDMS/DB database.

Runtime operations include the following:
Developing and running native CA IDMS/DB application programs (refer to the appropriate CA IDMS/DB documents)

Running Total and CICS application programs by using the runtime interface

Illustration of the Runtime Operations Phase

The following figure illustrates the runtime operations phase.

For more information on the runtime operations phase, see Runtime Operations Phase (see page 46).

Operating Environments Supported

The transparency option runs in the IBM z/OS and z/VSE operating environments; Batch and CICS operations are supported.

The standard CA IDMS interface requires a z/OS or z/VSE operating system at a version level that is supported by IBM. Additionally, for CICS clients, CA IDMS Version 18 requires functionality first introduced in the following CICS versions:

- CICS Transaction Server for z/OS V2.2
- CICS Transaction Server for z/VSE V1.1.1

CICS transactions which utilize the Total transparency option should not be defined as threadsafe.

Support for Total

Contents

- Total Features Supported (see page 18)
- Total Features Not Supported (see page 19)

This section provides information on Total features supported/not supported under the CA IDMS TOTAL Transparency.
Total Features Supported

The transparency supports most Total DML statements that access and update the database. Almost all Total DML statements supported by the transparency translate directly to corresponding CA IDMS DML statements. The Total DML statements described in the following table are translated to CA IDMS, but with some variations in function. The variations are explained in the table.

⚠️ Note: Total DML recovery and logging functions are not converted to CA IDMS/DB. CA IDMS/DB provides recovery utilities and journaling options that adequately replace these Total DML functions.

<table>
<thead>
<tr>
<th>For this Total DML statement</th>
<th>CA IDMS TOTAL Transparency does the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNTRL (purge)</td>
<td>Ignores the command</td>
</tr>
<tr>
<td>COMIT</td>
<td>Issues a COMMIT ALL</td>
</tr>
<tr>
<td>ENDLG</td>
<td>Issues a COMMIT</td>
</tr>
<tr>
<td>FREEX</td>
<td>Issues a COMMIT ALL</td>
</tr>
<tr>
<td>QMARK</td>
<td>Issues a COMMIT</td>
</tr>
<tr>
<td>QUIET</td>
<td>Issues a COMMIT</td>
</tr>
<tr>
<td>SINON and OPENX</td>
<td>Issues a BIND RUN UNIT statement</td>
</tr>
<tr>
<td></td>
<td>Issues BIND statements for all CA IDMS/DB record types described in the subschema</td>
</tr>
<tr>
<td></td>
<td>Issues a READY statement for all CA IDMS/DB database areas included in the subschema</td>
</tr>
<tr>
<td></td>
<td>After the runtime system processes a Total SINON or OPENX command, it will not reissue the CA IDMS/DB BIND/READY sequence.</td>
</tr>
<tr>
<td></td>
<td>When a SINON command is issued, the runtime interface performs the following actions:</td>
</tr>
<tr>
<td></td>
<td>Saves the name of the requesting task for use with a subsequent OPENX command</td>
</tr>
<tr>
<td></td>
<td>Loads the runtime control table (ETOTTBL) using the table name referenced by the DBMOD parameter of the SINON statement</td>
</tr>
<tr>
<td>CLOSX and SINOF</td>
<td>Issues a FINISH statement</td>
</tr>
<tr>
<td></td>
<td>FINISH releases all database areas (and therefore all database records) from program control. If a SINOF command is preceded by a CLOSX command, the runtime interface does not issue a FINISH in response to the SINOF command.</td>
</tr>
<tr>
<td>RQLOC (batch)</td>
<td>Accepts a key value from the Total application and returns a suitable DBKEY value. Note that for RQLOC, the module IDMSCALLC is required. If IDMSCALLC is not link edited to ETOTBINT, it is loaded the first time the RQLOC function is invoked.</td>
</tr>
</tbody>
</table>
Total Features Not Supported

The transparency's runtime interface does not support Total DML functions that are inappropriate in a CA IDMS/DB database environment. Information about unsupported functions is provided in the following table.

<table>
<thead>
<tr>
<th>Unsupported Total DML statement or function</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN DTO</td>
<td>These statements are related to recovery and logging. The runtime system returns an error for all of these statements. You can alter programs containing these statements in either of the following ways:</td>
</tr>
<tr>
<td>MARK 1</td>
<td>Check for the error status (FUNC)</td>
</tr>
<tr>
<td>RQLOC (CICS)</td>
<td>Remove the statement</td>
</tr>
<tr>
<td>RSTAT</td>
<td>Additionally, the logging options parameter of the SINON command and the <strong>REST</strong> parameter in Total DML commands are supported for retrieval only. When updating the database, these parameters are ignored.</td>
</tr>
<tr>
<td>WRITD</td>
<td>The <strong>REST</strong> parameter can be used in Total retrieval DML to indicate that the entire physical record (as defined in the DBMOD) is to be used, rather than just an element list of the record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SKIP option of Total RDNXT and FINDX</th>
<th>SKIP option of Total RDNXT and FINDX (for skipping unused records)</th>
</tr>
</thead>
</table>

| Some OPENX options                        | OPENX options that allow the user to ready Total files residing in the same area in different usage modes. The CA IDMS/DB database is readied for processing at the area level and each area can contain multiple record types. All CA IDMS/DB record types within a given area are accessed in the same usage mode. This approach differs from that of the Total Database Management System, which readies the database at the file level. Since each Total file contains records of only one type, each Total record type can be accessed in a different usage mode. |

<table>
<thead>
<tr>
<th>Some ADDV statements</th>
<th>ADDV statements issued for a coded variable record for which a record code and coded linkpath field were not defined in Total DDL. To override this restriction, you must modify the Total DDL used to generate the runtime control table prior to submitting the Total DDL to the transparency generator (see Step 2 -- Prepare Total DDL Statements on page ).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Do the following to define the coded variable file as a standard variable file:</td>
</tr>
<tr>
<td></td>
<td>1. Delete the Total DDL statement that defines the overlay element for the coded file</td>
</tr>
<tr>
<td></td>
<td>2. Define as a group item the data item designated as the overlay element for the coded file; the group item replacing the overlay element should reference all data items following the deleted statement</td>
</tr>
</tbody>
</table>
Phase: Data Description

The data description phase of transparency implementation is a planning and preparatory phase. It involves the following four steps:

- **Step 1 -- Select a Total Database** (see page 20)
- **Step 2 -- Prepare Total DDL Statements** (see page 20)
- **Step 3 -- Prepare the Physical CA IDMS/DB Database** (see page 21)
- **Step 4 -- Prepare the Logical CA IDMS/DB Database** (see page 21)

**Generator Input**

Later, during the database conversion phase, the statements you've prepared, the skeleton loader module (ETOTLOAD), and the skeleton runtime control table (ETOTTBL) are input to the generator. The generator uses this information to produce CA IDMS/DB schema and subschema descriptions and to produce both a customized loader and a customized runtime control table.

**Step 1 -- Select a Total Database**

To convert to a CA IDMS/DB environment, you must convert several databases. For the first conversion, it is recommended that you choose a small database (or, if necessary, a subset of a larger one) as a pilot project. This will allow you to gain experience, knowledge, and confidence for the full conversion.

**Step 2 -- Prepare Total DDL Statements**

After you select the Total database you want to convert, make sure you have all of the relevant Total DDL in a file. This file is part of the input to the generator.

If you are converting an entire database, you may already have the input file you need. If you are working with part of a database, however, you must prepare the input file appropriate to your conversion.

**Comments in the Total DDL**

If column 1 of a Total DDL input statement is blank, the generator considers the statement to be a comment and ignores it. This allows you to retain all of the original source DDL while selectively disabling some of the statements.

**Limits on the Total DDL**

At installation, the generator is set up to handle Total DDL input for up to 300 files, 5900 elements, 500 linkpaths, and 500 record codes. These limits are usually adequate for initial use of the transparency. If your DDL file exceeds these limits, see **Increasing Generator Input Limits** (see page 51) for information on increasing the limits.
Step 3 -- Prepare the Physical CA IDMS/DB Database

In this step you prepare physical CA IDMS/DB DDL statements. These statements describe the physical component of the CA IDMS/DB database. The physical component includes:

- **Segments** -- collections of files and areas
- **Database name table** -- names of databases and segments
- **Device-media control language (DMCL)** -- buffers, journals, identification of a database name table, identification of segments for inclusion in the database, file overrides, and area overrides.

**Note:** For more information on the syntax for CA IDMS/DB physical DDL statements, see the *CA IDMS Database Administration Section*. For information on the CA IDMS Command Facility which you use to submit physical DDL statements, see the *CA IDMS Common Facilities Section*.

Step 4 -- Prepare the Logical CA IDMS/DB Database

In this step you prepare generator control statements. These statements describe the logical component of the CA IDMS/DB database (the schema and subschema).

Detailed information for each generator control statement is presented in the next section.

The control statements and the Total DDL statements you prepared in step 2 are used to:

- Produce a complete schema description that reflects the Total database
- Produce complete subschema descriptions that reflect the Total database
- Build the loader module (ETOTLOAD); ETOTLOAD contains the information necessary to establish correspondence between the structure of the unloaded Total database and that of the CA IDMS/DB database
- Build a runtime control table (ETOTTBL); ETOTTBL replaces the Total DBMOD; ETOTTBL contains information that associates database requests in Total programs with the appropriate CA IDMS/DB data descriptions and relationships.

**Note:** After you code the control statements you can move on to the next transparency phase (see Database Conversion Phase (see page 34)).

Steps of the Data Description Phase
Generator Control Statements

Contents
- Summary of Statements (see page 22)
- Limits on Control Statements (see page 23)
- SYNONYMS Statement (see page 23)
- AREA Statement (see page 25)
- SCHEMA Statement (see page 26)
- SUBSCHEMA Statement (see page 28)
- USAGE-MODE Statement (see page 29)
- LOAD Statement (see page 32)

This section provides a table summarizing the generator control statements and a detailed description of each statement, including syntax and parameter descriptions.

Summary of Statements

The following table describes each generator control statement and specifies whether it is required or optional.

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNONYM</td>
<td>Opti</td>
</tr>
<tr>
<td></td>
<td>Assigns synonyms to Total element and file names. If this statement is not input, the generator assigns existing Total names to the corresponding CA IDMS/DB elements and records in the schema. To simplify the pilot project, we recommend that you not use synonyms.</td>
</tr>
<tr>
<td>AREA</td>
<td>Opti</td>
</tr>
<tr>
<td></td>
<td>Assigns Total files to user-specified CA IDMS/DB areas. If this statement is not included, the CA IDMS/DB record types that correspond to Total files will all be contained in the first area named in the SCHEMA statement.</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Req</td>
</tr>
<tr>
<td></td>
<td>Provides the generator with the SCHEMA, FILE, and AREA components of the CA IDMS/DB schema description.</td>
</tr>
<tr>
<td>SUBSCHEMA</td>
<td>Req</td>
</tr>
<tr>
<td></td>
<td>Provides the generator with the subschema description. The SUBSCHEMA statement allows you to name the subschema and to include informational entries in the subschema.</td>
</tr>
<tr>
<td>USAGE-MODE</td>
<td>Opti</td>
</tr>
<tr>
<td></td>
<td>Specifies CA IDMS/DB ready options for database areas that the runtime interface accesses in a usage mode other than shared update.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Opti</td>
</tr>
<tr>
<td></td>
<td>Specifies whether the unloaded Total data contains master file control keys and linkpath information, and specifies the logical record length of the unloaded Total data if it is other than 120 bytes. You need to include this statement only if the unloaded Total data will contain master file control keys and linkpath information, or if the logical record length of the unloaded Total data is other than 120 bytes.</td>
</tr>
</tbody>
</table>
Limits on Control Statements

Installation limits set on generator control statements are as follows:

- 2000 16-byte synonyms (files, records, sets)
- 1000 32-byte synonyms (elements)
- 2000 usage mode statements
- 500 schema/subschema statements

⚠️ **Note:** For more information on increasing the limit on schema and subschema statements (up to 5000), see Increasing Generator Input Limits (see page 51).

SYNONYMS Statement

The SYNONYMS statement allows you to take advantage of the longer element, record, and set names permitted under CA IDMS/DB.

Assigns Existing Total Name

When the transparency generates schema and subschema descriptions, it assigns the existing Total name to the corresponding CA IDMS/DB name. The Total file name is assigned to the CA IDMS/DB record type. The Total data item name is assigned to the CA IDMS/DB element.

Size Limits

The following are the size limits that apply to Total and CA IDMS/DB names:

- A Total file name allows 4 characters; the corresponding CA IDMS/DB record type name allows 16 characters
- A Total data item name allows 8 characters; the corresponding CA IDMS/DB element name allows 32 characters

With the SYNONYMS statement, you can override the transparency's assignment of Total element and file names to CA IDMS/DB elements and records.

You can also use the SYNONYMS statement to override the set names that the transparency assigns to CA IDMS/DB sets established to represent relationships between Total master and variable files. The transparency derives CA IDMS/DB set names from the applicable Total master file, linkpath, and variable file names.

Example of the Transparency Assigning a Set Name
The following example shows how a CA IDMS/DB set name is derived from Total master file, linkpath, and variable file names.

Master file is CUST + CUST
Linkpath is CO + CO
Variable file is CUOR + CUOR

CA IDMS/DB set name = CUST-CO-CUOR

Syntax

SYNONYMS

```
total-element-name idms-name
     IN total-file-name
     OF total-file-name
     transparency-assigned-set-name
```

Parameters

- **SYNONYMS**
  Required keyword; must occupy a line by itself and begin in column 1.

- **total-element-name/total-file-name/transparency-assigned-set-name**
  Specifies the Total element or file name or the transparency-derived set name for which a synonym is to be used; this clause must begin in column 8.

- **total-element-name**
  8-character element name specified in the input Total DDL.

- **total-file-name**
  4-character file name specified in the input Total DDL.

- **transparency-assigned-set-name**
  Set name derived by CA IDMS TOTAL Transparency for a set established to correspond to a Total master file/variable file relationship.

- **IN/OF total-file-name**
  Identifies the Total file for which total-element-name is defined in the Total DDL. This parameter must be specified if total-element-name is not unique to a given Total file (that is, if total-element-name exists for more than one file defined in the Total DDL) or if total-element-name does not begin with the 4-character name of the file for which it is defined.

- **idms-name**
  Specifies the CA IDMS/DB name to be used in place of the name supplied by the generator. CA IDMS/DB element names must be from 1 to 32 characters long, and CA IDMS/DB record and set names must be from 1 to 16 characters long.

Example

The following is an example of the SYNONYMS control statement.

```
SYNONYMS
  CUST       CUSTOMER
```

The following is an example of the SYNONYMS control statement.

```
SYNONYMS
  CUST       CUSTOMER
```
AREA Statement

The AREA statement allows you to specify the CA IDMS/DB database area(s) to which CA IDMS/DB records replacing Total files are assigned.

When generating schema record descriptions, the generator assigns each record type to the first area identified in the SCHEMA statement. The AREA statement permits you to override this default by naming specific Total files and the database areas to which corresponding CA IDMS/DB records are assigned.

Syntax

The following is the syntax for the AREA statement. You must specify the keyword AREA in column 1, idms-area-name in column 8, and total-file-name in column 12. The keyword and each variable must occupy an entire line.

```
AREA

idms-area-name  total-file-name
```

Parameters

- **AREA**
  Required keyword; must occupy a line by itself and begin in column 1.

- **idms-area-name**
  Specifies the name of a CA IDMS/DB database area to which CA IDMS/DB records corresponding to subsequently named Total files are assigned. Idms-area-name must be from 1 to 16 characters long, must occupy a line by itself, and must begin in column 8. Additionally, idms-area-name must be followed by at least one Total file name and must be named in the SCHEMA statement.

- **total-file-name**
  Specifies the name of a Total file. All occurrences of the CA IDMS/DB record type that correspond to this Total file are assigned to the database area indicated by idms-area-name. Total-file-name must be the name of a Total file defined in the input Total DDL. Total-file-name can be repeated as required, however each entry must occupy a line by itself, and must begin in column 12.

Example

The following is an example of the AREA control statement.

```
AREA

CUSTOMER-REGION
  CUST
  REMA
  SUOR
  ACRE
  INVE
  DARE
  ORNU
```
SCHEMA Statement

The generator SCHEMA statement describes the schema, files, and areas for CA IDMS/DB. This information is used to generate the CA IDMS/DB DDL SCHEMA, FILE, and AREA statements.

⚠️ Note: For more information on CA IDMS logical DDL, see the CA IDMS Database Administration Section.

Composed of Substatements

The generator SCHEMA statement is composed of these substatements:

- **SCHEMA** -- Names the schema and allows informational entries
- **AREA** -- Names CA IDMS/DB database areas and maps them into the named database files. To describe all areas in the CA IDMS/DB database, you can add as many areas as you need in AREA substatements.

Using the information included in the SCHEMA statement and (if provided) in the SYNONYMS and AREA statements the generator produces a CA IDMS/DB schema description that reflects the definition of the Total database defined in the input Total DDL. The schema that is produced includes complete definitions of the CA IDMS/DB record types that correspond to the Total files and describes the CA IDMS/DB sets that correspond to relationships among Total files.

The generator automatically generates the schema RECORD DESCRIPTION and SET DESCRIPTION components, converting Total file definitions to CA IDMS/DB record descriptions and Total master file-variable file relationships to CA IDMS/DB set descriptions. Any entries in the SYNONYMS control statement are applied to the appropriate record, element, and set descriptions. Any entries in the AREA statement are applied to the appropriate record descriptions.

Syntax

The following is the syntax for the SCHEMA statement. You must specify the keyword SCHEMA in column 1. All other clauses must begin between columns 8 and 11.

```
SCHEMA
  Schema name is schema-name
  Version is version-number
    NEXT  HIGhest
    LOWest
  PREPared by user-id
    REVIsed
  schema DEScription is description-text
```

Parameters

- **SCHEMA**
  Required keyword; must occupy a line by itself and begin in column 1.

- **SCHEMA NAME IS schema-name**
  Specifies the name of the schema produced by the generator. `Schema-name` must be a 1- to 8-character value. The first character must be #, $, @, or A through Z. The remaining characters can be #, $, @, A through Z, 0 through 9, or the hyphen (except as the last character or following another hyphen).

- **VERSION**
  Qualifies the schema with a version number. Version numbers must fall within the range 1 through 9999, whether specified explicitly or specified in relation to existing versions.

- **version-number**
  Specifies an explicit version number and must be an unsigned integer in the range 1 through 9999.

- **NEXT HIGHEST**
  Specifies the highest version number assigned to `schema-name` plus one. For example, if versions 3, 5, and 8 of schema ETOTSCHM exist in the dictionary, the following statement would result in version 9 of ETOTSCHM being added to the dictionary:

  SCHEMA NAME IS ETOTSCHM VERSION IS NEXT HIGHEST.

  **Note:** If NEXT is specified without HIGHEST or LOWEST, the schema compiler assumes NEXT HIGHEST.

- **NEXT LOWEST**
  Specifies the lowest version number assigned to `schema-name` minus one. For example, if versions 3, 5, and 8 of schema ETOTSCHM exist in the dictionary, the following statement would result in version 2 of ETOTSCHM being added to the dictionary:

  SCHEMA NAME IS ETOTSCHM VERSION IS NEXT LOWEST.

  **Note:** If NEXT is specified without HIGHEST or LOWEST, the schema compiler assumes NEXT HIGHEST.

- **PREPARED/REVISED BY user-name**
  Identifies the schema author. `User-name` can be any 1- to 32-character value. If the value includes spaces or delimiters, it must be enclosed in quotation marks. PREPARED/REVISED BY is informational only.
- **SCHEMA DESCRIPTION IS description-text**
  Specifies remarks concerning the schema. *Description-text* is a 1- to 40-character alphanumeric value. If it contains spaces or delimiters, it must be enclosed in quotation marks. SCHEMA DESCRIPTION is informational only.

- **MEMO DATE IS mm/dd/yy**
  Specifies the date on which the schema was created. MEMO DATE is informational only.

- **ADD AREA NAME IS idms-area-name**
  Specifies the name of a CA IDMS/DB database area. *Idms-area-name* must be a 1- to 16-character value. The characters can be #, $, @, A through Z, 0 through 9, or the hyphen (except as the first or last character or following another hyphen). At least one alphabetic or international symbol (#, $, @) is required. *Idms-area-name* must not be the same as the schema name or the name of any other component (including synonyms) within the schema. *Idms-area-name* is copied into DML programs and the DMCL, so do not use a keyword known to either the DMCL or the DML compiler. For a list of these keywords, see the *CA IDMS Database Administration Section*.

**Example**

The following is an example of the SCHEMA control statement.

SCHEMA
  SCHEMA NAME IS ETOTSCHM VERSION IS 1
  MEMO DATE IS 10/19/91.
  ADD
  AREA NAME IS CUSTOMER-REGION.

**SUBSCHEMA Statement**

The SUBSCHEMA statement names the subschema, associates it with a schema, and permits informational entries. The generator copies all area, record, and set descriptions included in the schema, providing access to the entire CA IDMS/DB database as defined in the schema. The generator also appends VALIDATE and GENERATE statements to produce a subschema load module. The subschema load module thus created is used by the runtime interface to satisfy database requests issued from Total application programs.

**Syntax**

The following is the syntax for the SUBSCHEMA statement. All clauses must begin between columns 8 and 11, with the exception of the OF SCHEMA NAME clause, which must be coded between columns 12 and 72.

SUBSCHEMA
  SUBschema name is subschema-name
  of SCHEMA name is schema-name
  Version is version-number
    HIGhest
    LOWest
Parameters

- **SUBSCHEMA**
  Required keyword; must occupy a line by itself and begin in column 1.

- **SUBSCHEMA NAME IS subschema-name**
  Identifies the source description of the subschema to the data dictionary. Subschema-name must be a 1- to 8-character value. The first character must be #, $, @, or A through Z. The remaining characters can be #, $, @, A through Z, 0 through 9, or the hyphen (except as the last character or following another hyphen). Subschema-name also becomes the CSECT name of the object subschema. If the object subschema subsequently is link edited alone for dynamic loading, subschema-name must be the load library member name.

- **OF SCHEMA NAME schema-name**
  Associates the named subschema with the generator-created schema. Schema-name must be the name of the schema identified in the SCHEMA statement.

- **VERSION version-number**
  Qualifies schema-name with a version number. Version-number must be an unsigned integer between 1 and 9999.

- **PREPARED/REVISED BY user-id**
  Identifies the user who is preparing the subschema. PREPARED/REVISED BY is informational only.

Example

The following is an example of the SUBSCHEMA control statement.

```
SUBSCHEMA
  SUBSCHEMA NAME IS ETOTSUBS OF SCHEMA ETOTSCHM.
```

**USAGE-MODE Statement**

Before performing database services for a given Total application program, the runtime interface readies each area defined in the generated subschema in shared update usage mode. The optional USAGE-MODE statement allows you to override the default usage mode for areas used by the Total task associated with the application program.

**Syntax**

Each clause of this statement (after the keyword USAGE-MODE) must occupy a line by itself and must begin in column 8. Include as many clauses as needed.

```
USAGE-MODE

  total-task-name  idms-area-name  usage-mode
  READY    ALL
```

**Expansion of usage-mode**


Parameters

- **USAGE-MODE**
  Required keyword; must occupy a line by itself and begin in column 1.

- **total-task-name/READY**
  Identifies the Total tasks that access the named area in the specified usage mode.

- **total-task-name**
  Specifies the name of the Total task as it exists in the SINON command issued by the Total application program.
  Total-task-name USAGE-MODE statements must precede any READY USAGE-MODE statements.

- **READY**
  Specifies that all Total tasks that do not match total-task-name are to be readied in the named usage mode.
  Total-task-name USAGE-MODE statements must precede any READY USAGE-MODE statements.

- **idms-area-name/ALL**
  Identifies the CA IDMS/DB areas affected by each USAGE-MODE statement.

- **idms-area-name**
  Identifies the CA IDMS/DB database area known to the subschema used by the runtime interface.

- **ALL**
  Specifies that the runtime interface is to issue a READY ALL DML verb, indicating to CA IDMS/DB that all areas are to be accessed in the specified usage mode. If you specify usage-mode (parameter explained hereafter) as DEFAULT, all areas are readied in the usage mode specified in the CA IDMS/DB subschema.

- **usage-mode**
  Identifies a valid CA IDMS/DB usage mode.

- **DEFault**
  Specifies that all areas are to be readied in the usage mode specified in the CA IDMS/DB subschema.

⚠️ **Note:** DEFAULT is valid only if ALL is specified; it is not used with idms-area-name.

- **SHAred**
  Allows other run units executing concurrently under the CA IDMS/DB central version (CV) either to ready the area in shared update or shared retrieval mode (SHARED UPDATE), or to ready the area in shared update, shared retrieval, protected update, or protected retrieval mode (SHARED RETRIEVAL).
- **PROtected**
  Prevents concurrent update of the area by run units executing under the same central version. Once a run unit has readied an area with the PROTECTED option, no other run unit can ready that area in any UPDATE usage mode until the first run unit releases it. A run unit cannot ready an area with the PROTECTED option if another run unit has readied the area in UPDATE usage mode or with the EXCLUSIVE option.

- **EXCLUSive**
  Prevents concurrent use of the area by any other run unit executing under the CA IDMS/DB central version. Once a run unit has readied an area with the EXCLUSIVE option, no other run unit can ready that area in any usage mode until the first run unit releases it.

- **RETrieval**
  Opens the area for retrieval only and allows other concurrently executing run units to open the same area in any usage mode other than one that is exclusive.

- **UPdate**
  Opens the area for both retrieval and update and allows other concurrently executing run units to open the same area in any usage mode other than one that is exclusive or protected.

**Usage**

**Mutually-Exclusive Clauses**

Do not use READY ALL and READY *idms-area-name* clauses in the same USAGE-MODE statement.

Do not use *total-task-name* ALL and *total-task-name* *idms-area-name* clauses in the same USAGE-MODE statement.

**READY ALL SHARED UPDATE**

The USAGE-MODE statement READY ALL SHARED UPDATE is the same as having no USAGE-MODE statements.

**Bypassing DML READY Verbs**

The USAGE-MODE statement READY ALL DEFAULT causes the runtime interface to bypass issuing any DML READY verbs. The areas needed for processing will be readied as specified in the subschema.

**Central Version Parameters**

The SHARED, PROTECTED, and EXCLUSIVE options apply only to programs running under the CA IDMS /DB central version.

**Examples**

**Example 1**

Suppose there are two tasks associated with an ETOTTBL and you wish to open the database areas associated with each task in different usage modes. USERPRG1 uses CA-IDMS-ADBEA1 and CA-IDMS-ADBEA2; USERPRG2 uses CA-IDMS-ADBEA2 and CA-IDMS-ADBEA3. Sample statements are listed as follows:
The areas for USERPRG1 are readied as specified in the USAGE-MODE table; for USERPRG2, CA-IDMS-ADBEA3 is readied as specified and CA-IDMS-ADBEA2 defaults to SHARED UPDATE. All areas in the subschema not readied by specific USAGE-MODE statements will be readied in SHARED UPDATE usage mode.

Example 2

Suppose you have many CA IDMS/DB areas in the subschema associated with an ETOTTBL. The task does not require access to all of the areas, so only required areas are readied. Sample statements are listed as follows:

<table>
<thead>
<tr>
<th>USAGE-MODE</th>
<th>USERPRG1</th>
<th>CA-IDMS-ADBEA1</th>
<th>RETRIEVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERPRG1</td>
<td>CA-IDMS-ADBEA2</td>
<td>PROTECTED UPDATE</td>
<td></td>
</tr>
<tr>
<td>USERPRG2</td>
<td>CA-IDMS-ADBEA3</td>
<td>EXCLUSIVE UPDATE</td>
<td></td>
</tr>
</tbody>
</table>

READY ALL DEFAULT

The results are the same as they are for example 1, except that any other areas in the CA IDMS/DB subschema will not be readied. Any task other than USERPRG1 or USERPRG2 will have no DML READY verb generated. The last statement in the example causes any other tasks to use the default ready mode as specified in the subschema.

LOAD Statement

If the unloaded Total data contains master file control keys and linkpath fields, use the LOAD statement to tell the loader so that these fields can be ignored when the corresponding CA IDMS/DB database is loaded. If master file control keys and linkpath fields are not present, the LOAD statement is optional.

Unloaded Total data includes master file control keys and linkpath fields only if a means other than the CSITULOD utility is used to unload the Total database.

Additionally, the LOAD statement includes a parameter to specify the logical record length of the unloaded Total data if this value is other than 120 bytes.

Syntax

LOAD

COMPACT

   ALL   LRECL logical-record-length

Parameters

- LOAD
  Required keyword; must occupy a line by itself and begin in column 1.
COMPACT/ALL
Specifies whether the unloaded Total data is to contain master file control keys and linkpath fields. Begin this clause in column 8.

COMPACT
Specifies that master records, standard variable records, and coded variable records unloaded from the Total database contain the file name in the first four bytes of the record, followed immediately by all data items. Linkpath fields are not present in an unloaded Total record, nor is the master file control key defined in Total DDL for master files. COMPACT is the default. It is informational only and specifies that the unloaded Total data will not contain these fields.

ALL
Specifies that unloaded Total master records, standard variable records, and coded variable records contain the information shown in the following table. Use ALL if you are using the CSITULOD utility to unload the Total database (specify COMPACT in exceptional cases only). If you use means other than CSITULOAD to unload the Total database, the contents of the unloaded Total records must conform to one of the options provided by this clause.

<table>
<thead>
<tr>
<th>Record Contents (s)</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>The first four bytes of each unloaded master record contain the master file name, followed by four bytes containing the internal reference point, followed by the master file control key, followed by the contents of each linkpath field (eight bytes per linkpath) defined for the master file, followed by all data items defined for the record.</td>
</tr>
<tr>
<td>Standard variable</td>
<td>The first four bytes of each unloaded standard variable record contain the standard variable file name, followed by the contents of the principal linkpath key for the associated master file, followed by all data items defined for the record.</td>
</tr>
<tr>
<td>Coded variable</td>
<td>The first four bytes of each unloaded coded variable record contain the coded variable file name, followed by two bytes containing the record code, followed by the contents of the principal linkpath key for the associated master file, followed by all data items defined for the record.</td>
</tr>
</tbody>
</table>

LRECL logical-record-length
Indicates the largest logical record length of unloaded Total data if this length is other than 120 bytes. Do not include LRECL if the logical record length is 120. You must specify the logical record length of the largest record if you are using variable-length records. Begin this clause in column 8.

Example
The following is an example of the LOAD control statement.
LOAD ALL
Database Conversion Phase

The process for converting a Total database to a CA IDMS/DB database involves the following steps:

1. Generate the data definitions and load program using the transparency generator ETOTMAIN

2. Compile the generated schema, using the schema compiler IDMSCHEM; this stores the schema description in the dictionary

3. Prepare the physical database description by:
   a. Submitting physical DDL statements to the CA IDMS Command Facility (IDMSBCF)
   b. Generating database name table and DMCL load modules
   c. Punching the database name table and DMCL load modules using the PUNCH utility statement
   d. Linking the resulting object modules to a load library

4. Prepare the subschema by:
   a. Compiling the generated subschema using the subschema compiler IDMSUBSC
   b. Punching the subschema load module
   c. Linking the subschema to a load library

5. Initialize the CA IDMS/DB database using the FORMAT utility statement

6. Unload the Total database using CSITULOD

7. Load the CA IDMS/DB database using the customized transparency loader ETOTLOAD or a user-written load program

These conversion steps are discussed in this section.

For more information on z/OS JCL, see z/OS Job Control Language (see page 52). For more information on z/VSE JCL, see z/VSE Job Control Language (see page 60).

- Step 1 Transparence Generator (ETOTMAIN) (see page 35)
- Step 2 Schema Preparation (see page 41)
- Step 3 Physical Database Preparation (see page 42)
- Step 4 Subschema Preparation (see page 43)
- Step 5 CA IDMS/DB Database Initialization (see page 45)
- Step 6 Total Database Unload Utility (see page 45)
- Step 7 Transparency Loader (ETOTLOAD) (see page 45)
Step 1 Transparency Generator (ETOTMAIN)

The first step in transparency conversion of a Total database to a CA IDMS/DB database is to execute the transparency generator ETOTMAIN.

Input to ETOTMAIN

You provide the following input to ETOTMAIN, in this order:

- Total DDL that defines the database being converted
- Control statements that describe the CA IDMS/DB database environment
- ETOTTBL skeleton
- ETOTLOAD skeleton

Output from ETOTMAIN

Output from ETOTMAIN includes:

- Schema source code
- Subschema source code
- Customized runtime control table
- Customized loader

For more information on messages issued by ETOTMAIN, see Generator Messages (see page 48).

Examples of Input and Output

Input -- Total DDL

The following is an example of Total DDL statements input to the transparency generator (ETOTMAIN).

```
BEGIN-DATA-BASE-GENERATION:
  DATA-BASE-NAME=CUORDABA  16:28:54  07/18/99  08/18/99
BEGIN-MASTER-DATA-SET:
  DATA-SET-NAME=CUST
  MASTER-DATA:
    CUSTROOT=8
    CUSTCTRL=6
    CUSTLKCO=8
    CUSTLKAR=8
    CUSTLKRE=8
    CUSTNAME=30
    CUSTADDR=30
    CUSTCIST=20
    *FILLER*=5
    SAMENAME=5
  END-DATA:
  LOGICAL-RECORD-LENGTH=128
END-MASTER-DATA-SET:
```
Output -- Schema Source

The following is an example of schema source statements generated by the transparency generator, ETOTMAIN. This output is based on the previous Total DDL example input.

ADD
SCHEMA NAME IS ETOTSCHM VERSION IS 1.

ADD
AREA NAME IS CUSTOMER-REGION.

ADD
RECORD NAME IS CUSTOMER
RECORD ID IS 101
LOCATION MODE IS CALC USING CUSTOMER-KEY
DUPLICATES ARE NOT ALLOWED
WITHIN AREA CUSTOMER-REGION.

02 CUSTOMER-KEY PIC X(6).
ADD RECORD NAME IS ORNU
RECORD ID IS 102
LOCATION MODE IS CALC USING ORNUCTRL
DUPLICATES ARE NOT ALLOWED
WITHIN AREA CUSTOMER-REGION.

02 ORNUCTRL PIC X(12).
  02 ORNU0001 PIC X(5).
  02 ORNU0002.
    03 ORNU0003 PIC X(2).
    03 ORNU0004.
      04 ORNU0005 PIC X(7).
      04 ORNU0006.
        05 ORNU0007 PIC X(1).
        05 ORNU0008 PIC X(5).
        04 ORNU0009.
        05 ORNU0010 PIC X(2).
        05 ORNU0011 PIC X(2).
  02 Filler

ADD RECORD NAME IS DARE
RECORD ID IS 103
LOCATION MODE IS CALC USING DARECTRL
DUPLICATES ARE NOT ALLOWED
WITHIN AREA CUSTOMER-REGION.

02 DARECTRL PIC X(4).
  02 SAME-NAME-OF-DARE PIC X(6).
  02 Filler

ADD RECORD NAME IS INVE
RECORD ID IS 104
LOCATION MODE IS CALC USING INVECTRL
DUPLICATES ARE NOT ALLOWED
WITHIN AREA CUSTOMER-REGION.

02 INVECTRL PIC X(6).
  02 INVEITDE PIC X(30).
  02 INVEITCO PIC X(5).
  02 INVEITPR PIC X(5).
  02 INVEQTOH PIC X(4).
  02 INVEQTOO PIC X(4).
  02 Filler

ADD RECORD NAME IS CUSTOMER-ORDER

RECORD ID IS 105
LOCATION MODE IS VIA CUST-TO-ORDER SET
WITHIN AREA CUSTOMER-REGION.

02 CUORCODE PIC X(2).
  02 CUORGORNU PIC X(12).
  02 CUORLINU PIC X(2).
  02 CUORDATA PIC X(55).
  02 CUOR-RECORD-HD REDEFINES CUORDATA.
    03 CUORDRDR PIC X(4).
    03 CUORDSOS PIC X(4).
    03 CUORCUST PIC X(6).
    03 CUORTOVA PIC X(5).
    03 CUORTOIT PIC X(2).
    03 CUORTERM PIC X(10).
    03 Filler PIC X(24).
  02 CUOR-RECORD-IT REDEFINES CUORDATA.
    03 CUORINVE PIC X(6).
    03 CUORQURE PIC X(4).
ADD RECORD NAME IS ACRE
RECORD ID IS 106
LOCATION MODE IS VIA CUST-AR-ACRE SET
WITHIN AREA CUSTOMER-REGION.

ADD RECORD NAME IS REMA
RECORD ID IS 107
LOCATION MODE IS VIA CUST-RE-REMA SET
WITHIN AREA CUSTOMER-REGION.

ADD SET NAME IS CUST-TO-ORDER
ORDER IS PRIOR
MODE IS CHAIN LINKED TO PRIOR
OWNER IS ORNU NEXT DBKEY POSITION IS 1
PRIOR DBKEY POSITION IS 2
MEMBER IS CUSTOMER-ORDER NEXT DBKEY POSITION IS 1
PRIOR DBKEY POSITION IS 2
MANDATORY AUTOMATIC.

ADD SET NAME IS DARE-DR-CUOR
ORDER IS PRIOR
MODE IS CHAIN LINKED TO PRIOR
OWNER IS DARE NEXT DBKEY POSITION IS 1
PRIOR DBKEY POSITION IS 2
MEMBER IS CUSTOMER-ORDER NEXT DBKEY POSITION IS 3
PRIOR DBKEY POSITION IS 4
MANDATORY MANUAL.

ADD SET NAME IS DARE-DS-CUOR
ORDER IS PRIOR
MODE IS CHAIN LINKED TO PRIOR
OWNER IS DARE NEXT DBKEY POSITION IS 3
MEMBER IS CUSTOMER-ORDER

ADD SET NAME IS CUST-CO-CUOR
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS CUSTOMER

ADD SET NAME IS INVE-CO-CUOR
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS INVE

ADD SET NAME IS CUST-AR-ACRE
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS CUSTOMER

ADD SET NAME IS DARE-DN-ACRE
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS DARE

ADD SET NAME IS DARE-DG-ACRE
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS DARE

ADD SET NAME IS CUST-RE-REMA
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS CUSTOMER

ADD SET NAME IS INVE-RE-REMA
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS INVE

VALIDATE.
Output -- Subschema Source

The following is a sample of a subschema that is generated by the transparency generator, ETOTMAIN. This output is based on the previous Total DDL example input.

ADD
SUBSCHEMA NAME IS ETOTSUBS OF SCHEMA NAME ETOTSCHM.
   ADD AREA CUSTOMER-REGION.
      ADD RECORD CUSTOMER.
      ADD RECORD ORNU.
      ADD RECORD DARE.
      ADD RECORD INVE.
      ADD RECORD CUSTOMER-ORDER.
      ADD RECORD ACRE.
      ADD RECORD REMA.
      ADD SET CUST-TO-ORDER.
      ADD SET DARE-DR-CUOR.
      ADD SET DARE-DS-CUOR.
      ADD SET CUST-CO-CUOR.
      ADD SET INVE-CO-CUOR.
      ADD SET CUST-AR-ACRE.
      ADD SET DARE-DN-ACRE.
      ADD SET DARE-DG-ACRE.
      ADD SET CUST-RE-REMA.
      ADD SET INVE-RE-REMA.
      VALIDATE.
      GENERATE.

Step 2 Schema Preparation

To compile the schema created by the transparency generator, execute the program IDMSCHEM.

Schema Compiler Execution

The following figure shows the execution of the schema compiler.

IDMSDB--Step 2: Schema Preparation

Input to IDMSCHEM

The input to IDMSCHEM is the source schema DDL output by the generator.
Output from IDMSCHEM

Output from IDMSCHEM is as follows:

- A description of the schema stored in the dictionary
- A schema DDL input listing, including embedded warning and error messages issued by IDMSCHEM

**Note:** For more information on messages issued by IDMSCHEM, see the *CA IDMS Messages and Codes Section*. For more information on how to use the extended options of the schema compiler, see the *CA IDMS Database Administration Section*.

### Step 3 Physical Database Preparation

To prepare the physical database, perform these steps:

1. Submit DDL statements that describe the physical database using the CA IDMS Command Facility (IDMSBCF); physical DDL statements include the SEGMENT, DBNAME TABLE, and DMCL statements; include a GENERATE clause to generate database name table and DMCL load modules

2. Punch the database name table and DMCL load modules using the PUNCH utility statement

3. Link the resulting object modules to a load library

4. Identify the DMCL to the runtime system:
   - Central version -- specify the global DMCL name in the #DCPARM macro which defines system startup parameters; incorporate the Total DMCL statements into the global DMCL using the MODIFY DMCL statement
   - Local mode -- specify the DMCL name in the SYSIDMS parameter file

**Note:** For more information on the physical DDL statements and SYSIDMS, see the *CA IDMS Common Facilities Section*. For more information on the CA IDMS Command Facility, see the *CA IDMS Common Facilities Section*. For more information on #DCPARM, see the *CA IDMS System Operations Section*.

### IDMSBCF Execution

The following figure shows execution of IDMSBCF, the program you use to compile physical DDL.
IDMSDB--Step 3: Physical Database Preparation

Input to IDMSBCF

As input to IDMSBCF, you use the DDL statements that describe these physical components of the database:

- Segments
- Database name table (dbname table)
- DMCL

Output from IDMSBCF

Output from IDMSBCF is as follows:

- DMCL and database name table load modules
- A physical DDL input listing, including embedded warning and error messages issued by IDMSBCF (for a description of these messages, see the CA IDMS Messages and Codes Section)

Step 4 Subschema Preparation

To prepare the subschema, you perform these steps:

1. Compile the subschema using IDMSUBSC
2. Punch a subschema load module

3. Link edit the load module to a load library

**IDMSUBSC Execution**

The following figure shows the execution of the subschema compiler IDMSUBSC.

![IDMSUBSC Execution Diagram]

**IDMSDB--Step 4: Subschema Preparation**

**Input to IDMSUBSC**

Input to IDMSUBSC is the subschema DDL created by the transparency generator.

**Output from IDMSUBSC**

Output from IDMSUBSC is:

- A source description of the subschema stored in the dictionary, including embedded warning and error messages issued by IDMSUBSC
- A subschema load module stored in the dictionary load area (DDLDCLOD), if the source input contains a GENERATE statement; the subschema in the dictionary load area is used by applications running under the central version

**Note:** For more information on the IDMSUBSC messages, see the *CA IDMS Messages and Codes Section*.

**Punching the Subschema Load Module**

To punch the subschema load module, you submit a PUNCH LOAD MODULE statement to the DDDL compiler (IDMSDDDL).

Output from IDMSDDDL is an object deck. z/VSE users must be sure to save this output for use in step 7 of this conversion.
Linking the Subschema Load Module

Input to the linkage editor is the object deck produced by IDMSDDDL.

Output from the linkage editor is a load module in the load library. The subschema in the load library is used by applications running in local mode and the transparency loader.

Step 5 CA IDMS/DB Database Initialization

To initialize the CA IDMS/DB database (format database pages), you submit the FORMAT utility statement through the CA IDMS Command Facility (IDMSBCF).

Note: For more information on FORMAT, see the CA IDMS Utilities Section. For more information on IDMSBCF, see the CA IDMS Common Facilities Section.

Step 6 Total Database Unload Utility

To unload the Total database, execute the Cincom utility, CSITULOD. Specify the appropriate parameters to preserve all pointers and unload all elements. Refer to the appropriate Total documentation for instructions on executing this utility.

If a means other than the CSITULOD utility is used to unload the Total database, ensure that the logical order of the database is preserved. Additionally, ensure that the content of all unloaded Total records conforms to that specified in the LOAD statement input to the transparency generator, see Phase: Data Description (see page 20).

Step 7 Transparency Loader (ETOTLOAD)

Compile, link edit, and execute the transparency customized loader, ETOTLOAD, or your own load program to load data into the CA IDMS/DB database. You must run the loader in local mode.

Note: For more information on the messages provided by the transparency loader, see Appendix C, Loader Messages.

The ETOTLOAD job stream consists of several steps.

Before Compiling ETOTLOAD

Review the file description (FD) of the UNLOAD-FILE and edit as necessary before you compile ETOTLOAD.

z/VSE users should also review the COBOL INPUT-OUTPUT SECTION ASSIGN clause and edit it as necessary.
Before Executing ETOTLOAD

Before executing ETOTLOAD you must change all set orders of PRIOR to NEXT. To do this, execute the schema compiler IDMSCHEM with the following input:

modify schema schema-name.
modify set set-name-1 order is next.
modify set set-name-2 order is next.
.. .
modify set set-name-n order is next.
validate.
regenerate all subschemas.

Modify the set order for all sets in all schemas. Punch and link the subschema load modules.

After Executing ETOTLOAD

After executing the loader you must restore all set orders to PRIOR. To do this, execute the schema compiler IDMSCHEM with the following input:

modify schema schema-name.
modify set set-name-1 order is prior.
modify set set-name-2 order is prior.
.. .
modify set set-name-n order is prior.
validate.
regenerate all subschemas.

Modify the set order for all sets in all schemas. Punch and link the subschema load modules.

Runtime Operations Phase

Before You Begin:

For related information, see the specific appendices.

- **z/OS Job Control Language (see page 52)** for z/OS JCL.
- **z/VSE Job Control Language (see page 60)** for z/VSE JCL.
- [Runtime Messages](https://docops.ca.com/display/IDMS19/Runtime+Messages) for a discussion of the codes returned to the Total application program running in the transparency environment.

**Note:** For more information on runtime operations, see the *CA IDMS System Operations Section*.

What To Do

To execute a Total application program with the CA IDMS TOTAL Transparency runtime interface, perform the steps below.

**Step 1: Assemble and Link Edit ETOTTBL Output**
Assemble and link edit ETOTTBL output generated by ETOTMAIN.

Perform this step once to store the control table in the load library for use with all Total application programs. It is not repeated for each Total application program that uses the runtime interface.

**Step 2: Modify the Total Application Program**

Modify the Total application program to remove any Total DML functions not supported by the runtime interface (see Total Features Supported (see page 18)).

**Step 3: Recompile the Total Application Program**

Recompile the Total application program if it has been modified.

**Step 4: Assemble the ETOTCINT Macro**

Assemble the ETOTCINT macro if CICS applications will use the transparency to access the CA IDMS/DB database.

The syntax for the ETOTCINT macro used to generate the ETOTCINT module is shown below. You only need to assemble this macro once. Subsequently, it can be used with all Total CICS applications to access the CA IDMS/DB database.

```
ETOTCINT CWADISP= cics-cwa-displacement
   ,NAME= ETOTCINT csect-name
```

**Parameters**

- **CWADISP=cics-cwa-displacement**
  Specifies the displacement, in bytes, of the transparency work area within the CICS CWA area. Specify the same value given to the CWADISP operand when you assembled the CICSOPT macro to create your CICSOPTS module.

- **NAME=ETOTCINT/csect-name**
  Specifies the name of the generated module. The default is ETOTCINT. If specified, `csect-name` must be a 1- to 8-character alphanumeric value.

**Step 5: Create Your CICS Interface Module**

Most parameters for the CICSOPT macro are described in the System Operations section. Parameters specific to the Total/T interface are described below.

**Syntax**

```
,ESCTOT= YES
   NO
,ESCTBL= total-control-table
   ETOTTBL
```

For more information on this topic, see Creating the CICS CA IDMS/DB Interface (IDMSINTC) (see page 57) for z/OS and Creating the CICS CA IDMS/DB Interface (IDMSINTC) (see page 57) for z/VSE.
Parameters

- **ESCTOT=YES/NO**
  Specifies whether the IDMSINTC interface supports TOTAL/T. YES should be specified. The default is NO.

- **ESCTBL=ETOTTBL/total-control-table**
  Specifies the name of the load module containing the control table generated by ETOTMAIN. The default is ETOTTBL.

**Step 6: Link Edit the Compiled Total Application**

Link edit the compiled Total application program with the following transparency and CA IDMS/DB modules:

- **ETOTBINT (batch only) or ETOTCINT (CICS only)** -- The runtime interface module that passes translated database calls to CA IDMS/DB. ETOTBINT is supplied as an object module with the transparency; ETOTCINT is supplied as a macro that must be assembled prior to runtime.

- **ETOTTRAN (batch only)** -- The runtime interface module (supplied in object form) that translates Total database calls to CA IDMS/DB database calls.

- **IDMS (batch only)** -- The CA IDMS/DB module that will receive database calls passed by ETOTBINT.

- **IDMSOPTI (batch only)** -- The (optional) CA IDMS/DB module that passes runtime information to the CA IDMS/DB central version that will handle database calls issued by the runtime interface of the transparency. IDMSOPTI is used to indicate the mode of operation.

**Step 7: Execute the Total Application**

Execute the Total application as per standard CA IDMS/DB runtime procedures.

---

**Generator Messages**

When the generator detects a processing error, the generator writes a message to an error report. On the report, the message follows the record that contains the error. An asterisk is printed under the first character of the entry causing the error or warning. An E, W, or F labels each message, as follows:

<table>
<thead>
<tr>
<th>Label</th>
<th>Status</th>
<th>Effect on processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Nonfatal error</td>
<td>Generator continues processing</td>
</tr>
<tr>
<td>W</td>
<td>Warning</td>
<td>Generator continues processing</td>
</tr>
<tr>
<td>F</td>
<td>Fatal error</td>
<td>Generator terminates processing</td>
</tr>
</tbody>
</table>

This section lists and describes error, warning, and fatal messages issued by the generator.

- **Error Messages 3 (see page 49)**
- **Warning Messages 3 (see page 51)**
- **Fatal Messages (see page 51)**
## Error Messages 3

<table>
<thead>
<tr>
<th>Message issued</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAD GROUP ELEMENT LENGTH</td>
<td>Error in Total DDL; the specified group-length must match the combined lengths of the individual items making up the group.</td>
</tr>
<tr>
<td>CARD BUFFER FULL (ETOTPARS)</td>
<td>More than 500 SYNONYM, AREA, SCHEMA, DEVICE-MEDIA, SUBSCHEMA, and USAGE-MODE statements; relink ETOTMAIN to include ETOTR002 (see Increasing Generator Input Limits (see page 51)).</td>
</tr>
<tr>
<td>IMPROP ERP ELEME NT NAME</td>
<td>Invalid Total DDL data name or keyword; entry must be <em>FILLER</em>, RECORD-CODE, or an 8-character data name.</td>
</tr>
<tr>
<td>IMPROP ERP FILE NAME</td>
<td>Invalid file name; entry must be four characters in length and alphanumeric.</td>
</tr>
<tr>
<td>IMPROP ERP INTEGER</td>
<td>Expecting an integer; entry is nonnumeric.</td>
</tr>
<tr>
<td>IMPROP ERP SYNONYM M</td>
<td>Invalid or missing CA IDMS/DB element, record, or set name in SYNONYMS control statement; entry must be alphanumeric.</td>
</tr>
<tr>
<td>IMPROP ERP SYNONYM M SOURCE</td>
<td>Invalid Total element or file name or transparency set name in SYNONYMS control statement; entry must be 4, 8, or 12 characters long.</td>
</tr>
<tr>
<td>IMPROP ERP SYNTAX</td>
<td>Invalid Total DDL syntax.</td>
</tr>
<tr>
<td>INVALID AREA NAME</td>
<td>Invalid or missing CA IDMS/DB area name in AREA or USAGE-MODE control statement; entry must be from 1 to 16 characters long and must be alphanumeric.</td>
</tr>
<tr>
<td>INVALID LOAD LRECL</td>
<td>Invalid logical record length in LRECL entry of LOAD control statement; entry must be greater than or equal to 1.</td>
</tr>
<tr>
<td>INVALID LOAD SYNTAX</td>
<td>Invalid LOAD statement syntax; entry following keyword LOAD must be ALL, ALL., or LRECL.</td>
</tr>
<tr>
<td>Message issued</td>
<td>Reason</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
</tr>
<tr>
<td>INVALID SUBSCHE NAME</td>
<td>Invalid or missing CA IDMS/DB subschema name in SUBSCHEMA NAME IS entry of SUBSCHEMA control statement; entry must be from 1 to 8 characters long.</td>
</tr>
<tr>
<td>INVALID TASK NAME</td>
<td>Invalid Total task name in USAGE-MODE control statement; entry must be 8 characters long and must be alphanumeric.</td>
</tr>
<tr>
<td>INVALID USAGE</td>
<td>Invalid CA IDMS/DB usage mode in USAGE-MODE control statement; entry must specify RETRIEVAL, UPDATE, SHARED RETRIEVAL, PROTECTED RETRIEVAL, EXCLUSIVE RETRIEVAL, SHARED UPDATE, PROTECTED UPDATE, or EXCLUSIVE UPDATE.</td>
</tr>
<tr>
<td>INVALID USE OF READY</td>
<td>A READY ALL USAGE-MODE control statement was detected after a READY area-name USAGE-MODE control statement had been found, or vice versa.</td>
</tr>
<tr>
<td>INVALID USE OF PGMNAME</td>
<td>A total-task-name USAGE-MODE control statement specifying an idms-area-name was detected after a total-task-name ALL statement had been found, or vice versa.</td>
</tr>
<tr>
<td>MISSING BEGIN-DATABASE GENERATION</td>
<td>The beginning statement of the Total DDL file is missing.</td>
</tr>
<tr>
<td>SYMBOL-TABLE FULL</td>
<td>The number of files plus the number of elements in the entire DDL is greater than the table size allowed in the transparency.</td>
</tr>
<tr>
<td>UNDEFINED AREA</td>
<td>Unrecognized CA IDMS/DB area name in USAGE-MODE control statement; entry must be defined in SCHEMA control statement.</td>
</tr>
<tr>
<td>UNDEFINED ED ELEMEN T</td>
<td>Unrecognized Total data-element name; entry must be defined in Total DDL.</td>
</tr>
<tr>
<td>UNDEFINED ED FILE</td>
<td>Unrecognized Total file name in Total DDL, SYNONYMS control statement, or AREA control statement. Any Total file name used as the first four characters of a Total file element name in Total DDL must also be defined in Total DDL; a Total file name used to qualify a Total element name in the SYNONYMS control statement must be defined in the Total DDL; a Total file name entered in the AREA control statement must be defined in the Total DDL.</td>
</tr>
<tr>
<td>UNDEFINED ED SET NAME</td>
<td>Unrecognized transparency set name in SYNONYMS control statement; entry must be the name of a CA IDMS/DB set established by the generator.</td>
</tr>
</tbody>
</table>
Warning Messages 3

<table>
<thead>
<tr>
<th>Message issued</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEAREST PRIME TOO LARGE FOR HASH TABLE</td>
<td>Insufficient table size.</td>
</tr>
</tbody>
</table>

Fatal Messages

<table>
<thead>
<tr>
<th>Message issued</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>REACHED EOF ON SKEL-FILE</td>
<td>Invalid or missing loader skeleton (ETOTLOAD); probable JCL error.</td>
</tr>
<tr>
<td>REACHED EOF ON DB-TABLES</td>
<td>Invalid ETOTTBL input; probable JCL error.</td>
</tr>
</tbody>
</table>

Loader Messages

The following messages are issued by the CA IDMS TOTAL Transparency loader.

<table>
<thead>
<tr>
<th>Message issued</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>??? FILE NOT FOUND: INPUT:in</td>
<td>The unloaded Total database includes a record (input-record-name) that references a Total file not known to the loader. Ensure that the loader is reading the proper Total data and that the Total database was unloaded correctly.</td>
</tr>
<tr>
<td>??? STRANGE FILE TYPE</td>
<td>The array built into the WORKING-STORAGE SECTION of the transparency loader contains an invalid file-type field.</td>
</tr>
<tr>
<td>??? NOT ENOUGH OWNER DESCRIPTOR RECORDS</td>
<td>A transparency buffer limit has been exceeded.</td>
</tr>
</tbody>
</table>

Increasing Generator Input Limits

You may need to increase transparency generator input limits to accommodate large database conversions. The transparency generator (ETOTMAIN) comes in two versions. These versions differ from each other in the capacity for input to the generator.

Limits apply to:

- Total DDL statements
- Transparency generator control statements
The default installation link-edit of the transparency generator (ETOTMAIN) includes ETOTROOT, which sets lower limits on input to the transparency generator. These limits accommodate most uses of the transparency.

If you are working with larger databases and require larger limits, relink ETOTMAIN with ETOTROO2.

Note that you should not relink ETOTMAIN unless necessary. ETOTROO2 will increase the memory requirements of ETOTMAIN. The following tables show the default limits and optional larger limits for the two types of input to the transparency generator.

### Limits on Total DDL Statements

<table>
<thead>
<tr>
<th>Item</th>
<th>Default limit</th>
<th>ETOTROO2 limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>300</td>
<td>1000</td>
</tr>
<tr>
<td>Elements</td>
<td>5900</td>
<td>9000</td>
</tr>
<tr>
<td>Linkpaths (=sets)</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Record codes</td>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>

### Limits on Transparency Generator Control Statements

<table>
<thead>
<tr>
<th>Item</th>
<th>Default limit</th>
<th>ETOTROO2 limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-byte synonyms (files, records, sets)</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>32-byte synonyms (elements)</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Usage mode statements</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Schema/subschema statements</td>
<td>500</td>
<td>5000</td>
</tr>
</tbody>
</table>

### z/OS Job Control Language

#### Generate (ETOTMAIN)

Use the following JCL to execute the transparency generator (ETOTMAIN):

```plaintext
ETOTMAIN (z/OS)
//GENERATE EXEC PGM=ETOTMAIN, REGION=2048K
//STEPLIB DD DSN=idms.dba.loadlib, DISP=SHR
// DD DSN=idms.loadlib, DISP=SHR
//SYSLST DD SYSOUT=A
```
CA IDMS - 19.0

//dcmsg DD DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSIDMS DD *
DMCL=etotdmcl
Input other SYSIDMS parameters, as required
//
//SYSIPT DD DSN=user.srclib(totalddl),DISP=SHR
// DD DSN=user.srclib(ctrlstmt),DISP=SHR
// DD DSN=yourHLQ.CAGJSRC(ETOTTBL),DISP=SHR
// DD DSN=yourHLQ.CAGJSRC(ETOTLOAD),DISP=SHR
//SYSPCH DD DSN=user.etotschm,UNIT=disk,DISP=(NEW,CATL),
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
//SYSPCH03 DD DSN=user.etotsubs,UNIT=disk,DISP=(NEW,CATL),
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
//SYSPCH04 DD DSN=user.etottbl,UNIT=disk,DISP=(NEW,CATLG,
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
//SYSPCH05 DD DSN=user.etotload,UNIT=disk,DISP=(NEW,CATL),
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)

⚠️ Note: The punched output should be directed to sequential files. Do not use partitioned datasets.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.dba.loadlib</td>
<td>Dataset name of the CA IDMS/DB load library containing the DMCL and database name table load modules</td>
</tr>
<tr>
<td>idms.loadlib</td>
<td>Dataset name of the CA IDMS/DB load library containing CA IDMS executable modules</td>
</tr>
<tr>
<td>dcmmsg</td>
<td>DDname of the system message area</td>
</tr>
<tr>
<td>idms.sysmsg.ddldcmsg</td>
<td>Dataset name of the system message area</td>
</tr>
<tr>
<td>etotdmcl</td>
<td>Name of the DMCL to be accessed at runtime; use the name of the global DMCL</td>
</tr>
<tr>
<td>Note:</td>
<td>For more information on the SYSIDMS parameters, see the CA IDMS Common Facilities Section.</td>
</tr>
<tr>
<td>user.srclib</td>
<td>Dataset name of the user source library</td>
</tr>
<tr>
<td>totalddl</td>
<td>Name of the library member containing Total DDL</td>
</tr>
<tr>
<td>yourHLQ.CAGJSRC</td>
<td>Dataset name of the CA IDMS/DB source library</td>
</tr>
<tr>
<td>ctrlstmt</td>
<td>Name of the library member containing user-supplied control statements</td>
</tr>
<tr>
<td>user.etotschm</td>
<td>Dataset name of the file containing the schema source generated by ETOTMAIN. Run this output through the schema compiler (IDMSCHEM).</td>
</tr>
<tr>
<td>user.etotsubs</td>
<td>Dataset name of the file containing the subschema source generated by ETOTMAIN. Run this output through the subschema compiler (IDMSUBSC).</td>
</tr>
<tr>
<td>user.etottbl</td>
<td>Dataset name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL. Run this output through the assembly and link edit process.</td>
</tr>
<tr>
<td>user.etotload</td>
<td>Dataset name of the file containing the customized loader generated by ETOTMAIN from the skeleton loader, ETOTLOAD. Run this output through the COBOL compile and link edit process.</td>
</tr>
</tbody>
</table>
Assembling and Link Editing ETOTTBL

Use the JCL below to assemble and link edit ETOTTBL.

ETOTTBL z/OS

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>disk</code></td>
<td>Symbolic device name of the disk file</td>
</tr>
<tr>
<td><code>etottbl</code></td>
<td>Name of the customized transparency ETOTTBL load module, which replaces the Total DBMOD module</td>
</tr>
</tbody>
</table>

Compiling and Link Editing ETOTLOAD

Use the following JCL to compile and link edit ETOTLOAD.

ETOTLOAD z/OS

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>yourHLQ.CAGJMAC</code></td>
<td>Dataset name of the CA IDMS/DB macro library</td>
</tr>
<tr>
<td><code>cics.maclib</code></td>
<td>Dataset name of the CICS macro library</td>
</tr>
<tr>
<td><code>disk</code></td>
<td>Symbolic device name of the disk file</td>
</tr>
<tr>
<td><code>etottbl</code></td>
<td>Name of the customized transparency ETOTTBL load module, which replaces the Total DBMOD module</td>
</tr>
<tr>
<td><code>zOS.maclib</code></td>
<td>Dataset name of the system macro library</td>
</tr>
<tr>
<td><code>idms.loadlib</code></td>
<td>Dataset name of the CA IDMS/DB load library</td>
</tr>
<tr>
<td><code>user.etottbl</code></td>
<td>Dataset name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL</td>
</tr>
</tbody>
</table>
Specifying an Object Library in the LINK Step

In the LIB DD statement, you can specify a CA IDMS/DB object library rather than a load library. If you do this you must include the following statement as part of SYSIN for the LINK step:

```
INCLUDE LIB(IDCSCALC, IDMSUTIL, IDMSALIO)
```

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys1.coblib</td>
<td>Dataset name of file containing COBOL support modules</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name</td>
</tr>
<tr>
<td>&amp;. &amp;cob.</td>
<td>Punch file containing object code from the COBOL compile step</td>
</tr>
<tr>
<td>user. etotload</td>
<td>Dataset name of the file containing the customized loader generated by ETOTMAIN</td>
</tr>
<tr>
<td>idms.loadlib</td>
<td>Dataset name of the CA IDMS/DB library containing CA IDMS executable modules</td>
</tr>
<tr>
<td>etotsubs</td>
<td>Name of the subschema, as specified in the transparency generator control statements</td>
</tr>
<tr>
<td>etotload</td>
<td>Name of the customized load program</td>
</tr>
</tbody>
</table>

Loading the Database with ETOTLOAD

Use the following JCL to execute ETOTLOAD. Output from this step is processed by the FASTLOAD utility.

**Note:** For more information on the FASTLOAD utility statement, see the *CA IDMS Utilities Section.*
ETOTLOAD z/OS

﻿// Use ETOTLOAD to create the load file for FASTLOAD
╱╱(unloaded Total database is converted to CA IDMS/DB format)
//*************************************************************
//DBL1 EXEC PGM=ETOTLOAD,REGION=512K
//STEPLIB DD DSN=idsms.dba.loadlib,DISP=SHR
//DD DSN=idsms.loadlib,DISP=SHR
//unload DD DSN=total.data,DISP=SHR
//SYSIDMS DD *
//DMCL=etotdmcl DBNAME=etotal
//*/
///ddmsg DD DSN=idsms.sysmsg.ddldcmsg,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSJNL DD DUMMY
//RELDCTL DD DSN=relcntl,DISP=(NEW,CATLG),UNIT=DISK,
//DCB=(RECFM=FB,LRECL=60,BLKSIZ=1200),
//SPACE=(TRK,(10,10),RLSE)
//SYS002 DD DSN=sortdbl1,DISP=(NEW,CATLG),UNIT=DISK,
//DCB=(RECFM=FB,LRECL=60,BLKSIZ=1200),
//SPACE=(TRK,(10,10),RLSE)
//SYS002 DD DSN=sort1,DISP=(NEW,CATLG),UNIT=DISK,
//SPACE=(TRK,1),DCB=BLKSIZ=80,VOL=SER=TECH01

idms.dba.loadlib Dataset name of the CA IDMS/DB library containing the DMCL and the database name table load modules
idms.loadlib Dataset name of the CA IDMS/DB library containing CA IDMS executable modules
unload DDname of the file containing data unloaded from the Total database (total.data)
total.data Dataset name of the file containing data unloaded from the Total database
etotdmcl SYSIDMS parameter specifying the name of the DMCL to be accessed at runtime
Note: For more information on the SYSIDMS parameters, see the CA IDMS Common Facilities Section.

etotal SYSIDMS parameter specifying the name of the database to be accessed at runtime
Note: For more information on the SYSIDMS parameters, see the CA IDMS Common Facilities Section.
dcmsg DDname of the CA IDMS system message area (DDLDCCMSG)
idms.sysmsg.ddldcmsg Dataset name of the CA IDMS system message area (DDLDCCMSG)
reldctl Dataset name for the file containing a control record with information on the subschema, DMCL, and segment to be used with FASTLOAD
disk Symbolic device name
nnnnnnn Volume serial number of the file
sortdbl1 Output file from ETOTLOAD to be used as input for FASTLOAD
sort1 Punch file for DBL1 step
Assembling the CICS Transparency Interface (ETOTCINT)

For more information on the ETOTCINT syntax, see Runtime Operations Phase (see page 46).

ETOTCINT z/OS

```plaintext
//EXEC HLASMCL
//ASM.SYSLIB DD DSN=yourHLQ.CAGJSRC,DISP=SHR
// DD DSN=cics.maclib,DISP=SHR
// DD DSN=yourHLQ.CAGJMAC,DISP=SHR
//ASM.SYSIN DD *
ETOTCINT macro statement
END
//LKED.SYSLMOD DD DSN=idms.loadlib(etotcint),DISP=SHR
```

<table>
<thead>
<tr>
<th>yourHLQ.CAGJSRC</th>
<th>Dataset name of the CA IDMS/DB source library</th>
</tr>
</thead>
<tbody>
<tr>
<td>cics.maclib</td>
<td>Dataset name of the CICS macro library</td>
</tr>
<tr>
<td>idms.loadlib</td>
<td>Dataset name of the CA IDMS/DB load library</td>
</tr>
<tr>
<td>etotcint</td>
<td>Name of the ETOTCINT module</td>
</tr>
</tbody>
</table>

Creating the CICS CA IDMS/DB Interface (IDMSINTC)

Initial Installation

When installing the CA IDMS TOTAL Transparency for 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 TOTAL 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 solution or as a single product during an ADDON install. The IDMSINTC load module will include all modules specifically required to run the TOTAL Transparency.

Modifying the CICSOPTS

If you need to reassemble CICSOPTS to change any installation options, edit the source in your CUSTOM.SRCLIB member CICSOPTS and the link statements in your CUSTOME.LNKLIB member IDMSINTC.

Important! Be sure PLT entries are created to execute IDMSINTC at CICS startup.
**Batch Application Program Link Edit**

Use this JCL to link edit the compiled Total application program with the ETOTBINT transparency module and the CA IDMS/DB modules IDMSOPTI and IDMS.

**BATCH APPLICATION PROGRAM LINK EDIT z/OS**

```
//LKEDTOTA EXEC PGM=HEWL,PARM=(XREF,LET,LIST),REGION=256K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSLMOD DD DSN=idsms.loadlib,DISP=SHR
//SYSLIB DD DSN=idsms.loadlib,DISP=SHR
//userlib DD DSN=user.loadlib,DISP=SHR
//SYSLIN DD *
   INCLUDE userlib(userprog)
   INCLUDE SYSLIB(ETOTBINT)
   INCLUDE SYSLIB(IDMS)
   INCLUDE SYSLIB(IDMSOPTI) Optional
ENTRY userprog
NAME userprog(R)
/*
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.loadlib</td>
<td>Dataset name of the CA IDMS/DB load library</td>
</tr>
<tr>
<td>disk</td>
<td>Symbolic device name of the linkage editor work file</td>
</tr>
<tr>
<td>user.loadlib</td>
<td>Dataset name of the user load library containing the batch application program</td>
</tr>
<tr>
<td>userlib</td>
<td>DDname of the user load library containing the batch application program</td>
</tr>
<tr>
<td>userprog</td>
<td>Name of the Total batch application program</td>
</tr>
</tbody>
</table>

**CICS Application Program Link Edit**

Use this JCL to link edit the compiled Total application program with the ETOTCINT macro.

**CICS APPLICATION PROGRAM LINK EDIT z/OS**

```
//LKEDTOTA EXEC PGM=HEWL,PARM=(XREF,LET,LIST),REGION=256K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSLMOD DD DSN=idsms.loadlib,DISP=SHR
//SYSLIB DD DSN=idsms.loadlib,DISP=SHR
//userlib DD DSN=user.loadlib,DISP=SHR
//SYSLIN DD *
   INCLUDE userlib(userprog)
   INCLUDE SYSLIB(etotcint)
   additional INCLUDE statements for CICS interface modules as required:
ENTRY userprog
NAME userprog(R)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.loadlib</td>
<td>Dataset name of the CA IDMS/DB load library</td>
</tr>
</tbody>
</table>
cics.loadlib  Dataset name of the Total CICS load library

disk        Symbolic device name of the linkage editor work file

etotcint    Name of the ETOTCINT module

user.loadlib Dataset name of the user load library containing the Total CICS application program

userlib     DDname of the user load library containing the CICS application program

userprog    Name of the CICS application program

### Runtime JCL

Use the following JCL to execute a Total application program under the central version.

**Runtime z/OS**

```plaintext
//userprog  EXEC PGM=userprog//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//     DD DSN=idms.loadlib,DISP=SHR
//sysctl     DD DSN=idms.sysctl,DISP=SHR
```

**program input, as required**

<table>
<thead>
<tr>
<th>idms.dba.loadlib</th>
<th>Dataset name of the CA IDMS/DB load library containing the DMCL and database name table load modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.loadlib</td>
<td>Dataset name of the CA IDMS/DB load library containing CA IDMS executable modules</td>
</tr>
<tr>
<td>idms.sysctl</td>
<td>Dataset name of the SYSCTL file</td>
</tr>
<tr>
<td>sysctl</td>
<td>DDname of the SYSCTL file</td>
</tr>
<tr>
<td>userprog</td>
<td>Name of the Total application program</td>
</tr>
</tbody>
</table>

To execute Total application programs in local mode, remove the SYSCTL DD statement and insert the following statements after the STEPLIB DD statement:

```plaintext
//userdb        DD DSN=user.userdb,DISP=SHR
```

**additional database file assignments, as required**

<table>
<thead>
<tr>
<th>sysjrnl</th>
<th>DD DSN=idms.tapejrnl,DISP=(NEW,KEEP),UNIT=tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdb</td>
<td>DDname of the CA IDMS/DB database file</td>
</tr>
<tr>
<td>user.userdb</td>
<td>Dataset name of the CA IDMS/DB database file</td>
</tr>
<tr>
<td>sysjrnl</td>
<td>DDname of the tape journal file</td>
</tr>
<tr>
<td>idms.tapejrnl</td>
<td>Dataset name of the tape journal file</td>
</tr>
<tr>
<td>tape</td>
<td>Symbolic device name of the tape journal file</td>
</tr>
</tbody>
</table>
The following is the JCL you use to run the transparency generator (ETOTMAIN).

**ETOTMAIN (z/VSE)**

```plaintext
// LIBDEF *,SEARCH=idms.sublib
// DLBL dcmgs,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT SYSnnn,nnn
// ASSIGN SYSnnn,DISK,VOL=nnnnnn,SHR

* // DLBL IDMSPCH,'user.etotschm',1,SD // EXTENT SYSnnn,nnn,ssss,lill // DLBL SYSPC03,'user.etotsubs',1,SD // EXTENT SYSnnn,nnn,ssss,lill

// DLBL SYSPC04,'user.etottbl',1,SD // EXTENT SYSnnn,nnn,ssss,lill // DBL SYSPC05,'user.etotload',1,SD // EXTENT SYSnnn,nnn,ssss,lill

// ASSIGN SYSnnn,DISK,VOL=nnnnnn,SHR

* // DLBL SYSIDMS,'#SYSIPT' // EXEC ETOTMAIN,SIZE=448K DMCL=etotdmcl FILENAME=SYSPC03 BLKSIZE=80 FILENAME=SYSPC04 BLKSIZE=80 FILENAME=SYSPC05 BLKSIZE=80

Input other SYSIDMS parameters, as required

*/
Total DDL source statements
User supplied control statements
ETOTTBL source statements
ETOTLOAD source statements
*/
```

<table>
<thead>
<tr>
<th>idms.</th>
<th>The name of the CA IDMS library</th>
</tr>
</thead>
<tbody>
<tr>
<td>sublib</td>
<td></td>
</tr>
<tr>
<td>dcmgs</td>
<td>Filename of the system message area (ddldcmsg)</td>
</tr>
<tr>
<td>idms.</td>
<td>File ID of the CA IDMS system message area (DDLDCMSG)</td>
</tr>
<tr>
<td>sysmsg.</td>
<td></td>
</tr>
<tr>
<td>ddldcmsg</td>
<td></td>
</tr>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number</td>
</tr>
<tr>
<td>user.</td>
<td></td>
</tr>
<tr>
<td>etotschm</td>
<td>Dataset name of the file containing the schema source generated by ETOTMAIN. Run this output through the schema compiler (IDMSCHEM).</td>
</tr>
<tr>
<td>user.</td>
<td></td>
</tr>
<tr>
<td>etotsubs</td>
<td>Name of the file containing the subschema source generated by ETOTMAIN. Run this output through the subschema compiler (IDMSUBSC).</td>
</tr>
<tr>
<td>ssss</td>
<td>Starting track (CKD) or block (FBA) of the disk extent</td>
</tr>
<tr>
<td>lill</td>
<td>Number of tracks (CKD) or blocks (FBA) in the disk extent</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number</td>
</tr>
</tbody>
</table>
Assembling and Link Editing ETOTTBL 1

Use the following JCL to assemble and link edit ETOTTBL.

**ETOTTBL (z/VSE)**

```
// LIBDEF *,CATALOG=user.sublib// DLBL IJSYSIN,'user.etottbl',1,SD
// EXTENT SYSIPT,nnnnn
ASSGN SYSIPT,DISK,VOL=nnnnnn,SHR
// OPTION CATAL,NODECK,NOSYM
PHASE etottbl,*
// EXEC ASMA90
/*
// EXEC LNKEDT
/*
CLOSE SYSIPT,SYSRDR
```

** user.** The name of the user sublibrary

** sublib**

** user.** Name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL. Run this output through the assembly and link edit process.

** etottbl**

** nnnn** Volume serial number

** nn**

** etottbl** Name of the customized transparency ETOTTBL load module, which replaces the Total DBMOD module

Compile and Link Edit ETOTLOAD

Use the following JCL to compile and link edit ETOTLOAD.

**ETOTLOAD (z/VSE)**

```
// LIBDEF *,SEARCH=idms.sublib,CATALOG=user.sublib// DLBL IJSYSIN,'user.etotload',1,SD
// EXTENT SYSIPT,nnnnnn
```
Loading Database with ETOTLOAD

Use the following JCL to execute ETOTLOAD. Output from this step is processed by the FASTLOAD utility.

Note: For more information on the FASTLOAD utility statement, see the CA IDMS Utilities Section.

ETOTLOAD (z/VSE)

* // LIBDEF *,SEARCH=(user.sublib,idms.sublib)
// DLBL dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT SYSnnn,nnnnn
// ASSGN SYSnns,DISK, VOL=n nn nn n,SHR
* // DLBL DMCLCH,'sort1',0,SD
// EXTENT SYSnnn,nnnnn,,ssss,1
// DLBL SYSO02,'sortdb1',0,SD
// EXTENT SYSnnn,nnnnn,,ssss,1111
// DLBL REDCTL,'redctl',0,SD
// EXTENT SYSnnn,nnnnn,,ssss,5
// ASSGN SYSnnn,DISK, VOL=n nn nn n,SHR
* // DLBL unload,'total.data',0,SD
// EXTENT SYSnnn,nnnnn,,ssss,1111
// ASSGN SYSnns,DISK, VOL=n nn nn n,SHR
* // DLBL SYSDMS,'#SYSIPT'
// EXEC ETOTLOAD,SIZE=256K
DMCL=etotdmcl DBNAME=etotal
/*
*/

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idms.sublib</td>
<td>The name of the CA IDMS library</td>
</tr>
<tr>
<td>user.sublib</td>
<td>The name of the user sublibrary</td>
</tr>
<tr>
<td>dcmgs</td>
<td>Filename of the system message area (ddldcmsg)</td>
</tr>
<tr>
<td>idms.sysmsg.</td>
<td>File ID of the CA IDMS system message area (DDLDCMSG)</td>
</tr>
<tr>
<td>ddldcmsg</td>
<td></td>
</tr>
<tr>
<td>SYSnnn</td>
<td>Logical unit of the volume for which the extent is effective</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number of the file</td>
</tr>
<tr>
<td>sort1</td>
<td>Punch file for DBL1 step</td>
</tr>
<tr>
<td>sortdb11</td>
<td>Output file from ETOTLOAD to be used as input for FASTLOAD</td>
</tr>
<tr>
<td>reldct1</td>
<td>File ID for the file containing a control record with information on the subschema, DMCL, and segment to be used with FASTLOAD</td>
</tr>
<tr>
<td>nnnnnn</td>
<td>Volume serial number</td>
</tr>
<tr>
<td>ssss</td>
<td>Starting track (CKD) or block (FBA) of the disk extent</td>
</tr>
<tr>
<td>lll</td>
<td>Number of tracks (CKD) or blocks (FBA) in the disk extent</td>
</tr>
<tr>
<td>unload</td>
<td>Filename of the file containing data unloaded from the Total database (total.data)</td>
</tr>
<tr>
<td>total.data</td>
<td>File ID of the file containing data unloaded from the Total database</td>
</tr>
<tr>
<td>etotdmcl</td>
<td>SYSIDMS parameter specifying the name of the DMCL to be accessed at runtime</td>
</tr>
<tr>
<td>Note:</td>
<td>For more information on the SYSIDMs parameters, see the CA IDMS Common Facilities Section.</td>
</tr>
<tr>
<td>etotal</td>
<td>SYSIDMS parameter specifying the name of the database to be accessed at runtime</td>
</tr>
<tr>
<td>Note:</td>
<td>For more information on the SYSIDMs parameters, see the CA IDMS Common Facilities Section.</td>
</tr>
</tbody>
</table>

Assembling the CICS/Transparency Interface (ETOTCINT)

The following is the JCL you use to assemble the CICS/transparency interface (ETOTCINT).

⚠️ Note: For more information on the ETOTCINT syntax, see the Runtime Operations Phase (see page 46).

**ETOTCINT (z/VSE)**

```plaintext
* // LIBDEF *,SEARCH=(idms.sublib) // DLBL IJSYSPH,'user.txtfile',0 // EXTENT SYSPCH,nnnnnn,,sss,lli ASSIGN SYSPCH,DISK,PERM, VOL=nmmnnn,T SHR // OPTION DECK,NOLINK,NOEDECK,LIST,NORLD,NOXREF```
Creating the CICS CA IDMS/DB Interface (IDMSINTC) 1

Initial Installation

When installing the CA IDMS TOTAL Transparency for 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 TOTAL 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 base install or as a single product during ADDON install. The IDMSINTC phase will include all of the modules specifically required to run the TOTAL Transparency.

Modifying CICSOPTS

If you need to reassemble CICSOPTS to change any installation options, you can use the source in your CUSTOM.SRCLIB member CICSOPTS and the link statements in your CUSTOM.LNKLIB member IDMSINTC.

Important! Be sure PLT entries are created to execute IDMSINTC at CICS startup.

Note: For information on the CICSOPTS macro and its parameters, see the CA IDMS System Operations Section.
Batch Application Program Link Edit 1

Use the JCL below to link edit a batch application program.

**Batch Application Program Link Edit (z/VSE)**

```
// LIBDEF *,SEARCH=(idms.sublib),CATALOG=user.sublib
// OPTION CATAL
PHASE userprog,*
INCLUDE userprog
INCLUDE ETOTBINT
INCLUDE IDMS
INCLUDE IDMSOPTI  Optional
ENTRY userprog
// EXEC LNKEDT
/*
```

<table>
<thead>
<tr>
<th>idms.sublib</th>
<th>The name of the CA IDMS library</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.sublib</td>
<td>The name of the user sublibrary</td>
</tr>
<tr>
<td>userprog</td>
<td>Name of the Total batch application program</td>
</tr>
</tbody>
</table>

CICS Application Program Link Edit 1

Use the following JCL to link edit a CICS application program.

**CICS Application Program Link Edit (z/VSE)**

```
// LIBDEF *,SEARCH=(idms.sublib),CATALOG=user.sublib// OPTION CATAL
PHASE userprog,*
INCLUDE userprog
INCLUDE etotcint
Additional INCLUDES for CICS interface modules, as required
ENTRY userprog
// EXEC LNKEDT
/*
```

<table>
<thead>
<tr>
<th>idms.sublib</th>
<th>The name of the CA IDMS library</th>
</tr>
</thead>
<tbody>
<tr>
<td>user.sublib</td>
<td>The name of the user sublibrary</td>
</tr>
<tr>
<td>userprog</td>
<td>Name of the Total CICS application program</td>
</tr>
<tr>
<td>etotcint</td>
<td>Name of the ETOTCINT module</td>
</tr>
</tbody>
</table>

JCL Runtime

Use the following JCL to execute a Total application program under the central version.

**Runtime (z/VSE)**

```
// LIBDEF *,SEARCH=(user.sublib,idms.sublib)
// DLBL sysctl,'idms.sysctl',1999/365,SD
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=nnnnnnn,SHR
```
// DLBL SYSIDMS,'#SYSIPT'
// EXEC userprog
Optional SYSIDMS parameters
/*
Program input if any
*/

user.sublib    The name of the user sublibrary
idms.sublib    The name of the CA IDMS library
sysctl         Filename of the SYSCTL file
idms.sysctl    File ID of the SYSCTL file
SYSnnn         Logical unit of the volume for which the extent is effective

nnnnnnn        Volume serial number
userprog       Name of the Total application program
SYSIDMS
parameters     Parameters you specify to establish a runtime environment.

Note: For more information on the SYSIDMS parameters, see the CA IDMS Common
Facilities Section.

Executing in Local Mode

To execute user application programs in local mode, remove the SYSCTL statements, and insert the
following statements before the EXEC userprog statement:

// DLBL  dcmgs,'idms.sysmsg.ddldcmgs',1999/365,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=nnnnnnn,SHR
// DLBL  userdb,'user.userdb',,DA
// EXTENT  SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=nnnnnnn,SHR
Additional database file assignments, as required
// TLBL  sysjrnl,'idms.tapejrnl',,nnnnnnn,,f
// ASSGN  sys009,TAPE,VOL=nnnnnnn

dcmgs          Filename of the system message area (ddldcmgs)
idms.sysmsg.ddldcmgs File name of the CA IDMS system message area (DDLDCMSG)
userdb         Filename of the user CA IDMS/DB database file
user.userdb    File ID of the user CA IDMS/DB database file
SYSnnn         Logical unit assignment of the user CA IDMS/DB database file
sysjrnl        Filename of the tape journal file
sys009         Logical unit assignment of the tape journal file
idms.tapejrnl  File ID of the tape journal file

nnnnnn        Volume serial number
f              File number of the tape journal file