CA Ideal™ for CA Datacom®

Creating Dataviews Guide

Version 14.02

CA Technologies Product References

This document references the following CA products:

- CA Datacom® CICS Services
- CA-Datacom® Datadictionary
- CA Datacom®/DB
- CA Datacom® SQL
- CA Ideal™ for Datacom® (CA Ideal)
- CA Ideal™ for DB2
- CA Ideal™ for VSAM

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction

This guide describes how to create and use dataviews using mainframe CA Ideal. Included are commands and functions specific to dataviews. For more information about descriptions and complete syntax of these functions and all CA Ideal commands, see the *Command Reference Guide*.

With Dataview Administration authorization level or higher, you can create or modify a database. All users can display a dataview.

Your View of Data

The CA Ideal user perceives the data as a named collection of columns or fields in a table. The applications developer can view each row or entry in this table as if it were a physical row or a record.

The following table represents data related to employees in an organization:

EMP-NO	**EMP-NAME (Last First Middle)**			**STREET**
5735	AARON	PAUL	T	1643 Hillside Road
6322	DOBBS	JOHN	I	123 Moyamensing Ave
4789	GOLD	SUE	M	222 Morning Glory Lane

In the Program Definition Language (PDL), there are statements to process the data as if it were an actual collection of records with references made to any field by the name of that field. From the user's standpoint, you can find any row (record) or set of rows by using the PDL FOR statement in the procedure definition in the *Command Reference Guide*. For example, you can find the second row by using a FOR construct that contains the clause WHERE
EMP-NO = '6322'. After you are positioned on a given row in the EMP-HIST dataview, you can reference or update the values DOBBS and 6322 by specifying the column names LAST-NAME and EMP-NO, respectively. In reality, these fields could comprise an entire record of a table or they could be a set of noncontiguous fields.

For SQL Access

With SQL access, you can process data by either Procedure Definition Language (PDL) statements or embedded SQL Data Manipulation Language (DML) statements. The data is treated as an actual collection of rows, with references made to any column by that column's name. In reality, these columns could comprise an entire row of a table or they could be a set of non-contiguous columns as defined in a view.

Note: SQL tables follow normalized rules and do not have group fields, occur, or redefines. SQL conventions do not support the hyphen; underscore is used instead.

General Description

A dataview is a logical view of data that makes requests of the data, independently of the storage structure. A dataview can be shared across many applications. A CA Ideal application only needs to name an existing dataview in the Program Resource Table and reference the various columns or fields in the Program Procedure. All attributes of the columns or fields are bound to the CA Ideal program at compile time, based on the status of components of the dataview when the dataview was cataloged.

For CA Datacom/DB only, the same table may be accessible using SQL and native access. Separate dataviews are required for each method, and these may present different formats for the same data (e.g. native access allows group fields, SQL does not). CA Ideal will present the program with a consistent form of the individual data fields regardless of the access used, even for types such as dates.

Mainframe CA Ideal supports four types of dataviews as follows:

CA Ideal SQL access dataviews

Dataviews for SQL objects (tables, views, or synonyms to access the CA Datacom SQL) to use as resources for CA Ideal programs. SQL access dataviews can be for DB2 or CA Datacom SQL access and are created with the CATALOG command. The SQL objects are created and maintained using database facilities.

CA Datacom/DB native command access dataviews

Dataviews for CA Datacom/DB native access tables to use as resources for CA Ideal programs. These dataviews are modeled, created, and maintained in CA Datacom/DB.

Sequential file dataviews

Dataviews that refer to sequential data files to use as resources for CA Ideal programs. These dataviews can be modeled in the dictionary or created and maintained in CA Ideal using fill-in screens to describe file and field attributes. When the dataview is created and maintained in CA Ideal, it is referred to as an unmodeled dataview because it is not modeled in the dictionary.

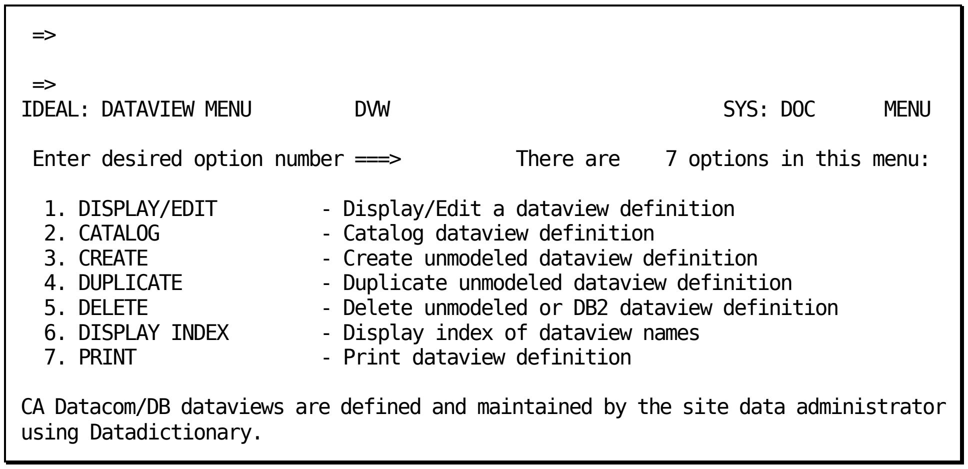
VSAM file dataviews

Dataviews that refer to VSAM files to use as resources for CA Ideal programs. These are unmodeled dataviews, created and maintained in CA Ideal using fill-in screens to describe file and field attributes.

Dataview Menu

The functions that CA Ideal offers for displaying or printing dataviews and for displaying a list of all dataview occurrences are illustrated in the Dataview Menu that follows. To access this menu, select option 2 on the Main Menu or enter the CA Ideal command DATAVIEW. To return to the Main Menu, press the Clear key.

Note: If a dataview is related to a system, the user must be authorized to that system to print or display the dataview or to run programs that have this dataview as a resource.



```
 =>

 =>
IDEAL: DATAVIEW MENU         DVW                              SYS: DOC      MENU

 Enter desired option number ===>         There are    7 options in this menu:

  1. DISPLAY/EDIT         - Display/Edit a dataview definition
  2. CATALOG              - Catalog dataview definition
  3. CREATE               - Create unmodeled dataview definition
  4. DUPLICATE            - Duplicate unmodeled dataview definition
  5. DELETE               - Delete unmodeled or DB2 dataview definition
  6. DISPLAY INDEX        - Display index of dataview names
  7. PRINT                - Print dataview definition

CA Datacom/DB dataviews are defined and maintained by the site data administrator
using Datadictionary.
```

The following functions are available from the Dataview Menu:

DISPLAY/EDIT

Select Option 1 to display or edit a dataview definition. You are prompted to specify whether you want to display the dataview or edit it.

CATALOG

Select Option 2 to catalog a dataview definition. A prompter panel displays to help you complete the command.

CREATE

Select Option 3 to create a VSAM or sequential dataview definition. The Dataview Identification fill-in displays.

DUPLICATE

Select Option 4 to duplicate a VSAM or sequential file dataview definition. A prompter displays to request the new name for the duplicated definition.

DELETE

Select Option 5 to delete a VSAM, SQL, or sequential file dataview definition.

DISPLAY INDEX

Select Option 6 to display a list of cataloged dataviews.

PRINT

Select Option 7 to print a cataloged dataview definition.

For more information about command syntax equivalent of each menu selection, see the *Command Reference Guide*. If you enter an incomplete command, a prompter displays to request the missing information.

After a fill-in is complete:

- Press any function key except CLEAR, PA1 (Reshow), and PA2 (display PF/PA key assignments) to apply the data.
- Press the Enter key to apply the data and leave the current fill-in displayed.
- Enter the appropriate command or press the appropriate PF key to continue with the next fill-in.

Cataloging a Dataview

A dataview must be cataloged with the CATALOG DATAVIEW command before CA Ideal can access it. When it is cataloged, CA Ideal captures all information about the dataview, including all column information. This information is incorporated into each program that uses the dataview as a resource when the program is compiled, and actually becomes part of the program object module.

If any change is made to the file structure that affects the dataview (for example, a change in the element length) or if a change is made that affects the attributes of any column in the dataview (such as class, type, length, decimal positions, or the relative position of the column in any element), then you must recatalog the dataview and recompile all programs using that dataview in CA Ideal.

To catalog a dataview, use the CATALOG command or select option 2 (CATALOG) from the Dataview Menu. The syntax of the CATALOG command is shown below.

Note: Access the CATALOG DATAVIEW prompter by selecting option 3 on the CA Ideal Administration Maintenance Menu.

This command has the following format:

```
        {*                              }
CATALOG {[dbms] DATAVIEW dvw-identifier }
        {                               }
             [[WITH] {VALIDATION   } ]
             [       {NOVALIDATION } ]
             [       {             } ]
```

***** An asterisk represents the current dataview and can be substituted for the DATAVIEW phrase.

dbms (Mixed SQL site only)

The database management system that SQL statements using this dataview access. This option overrides the DBMS specified by a SET ENVIRONMENT SQL command.

Valid values are:

DB-The dataview accesses an CA Datacom SQL object.

DB2-The dataview accesses a DB2 object.

Note: You can use this option only for cataloging a new dataview, not for recataloging an existing dataview. To change the DBMS for a dataview that was already cataloged, you must delete the dataview and catalog again.

dvw-identifier

Identifies the dataview. The dvw-identifier includes different parts, depending on the type of dataview to identify:

For SQL dataviews-Specify the dvw-identifier as:

authid.dvw-name

authid-The one- to eight-character authorization ID required for SQL dataviews only.

dvw-name-The 1- to 18-character name of the dataview. For SQL dataviews, the dataview name is the name of an SQL object (table, view, or synonym).

For CA Datacom/DB native access, sequential, and VSAM dataviews-Specify the *dvw-identifier* as:

***dvw-name* [VERSION *ver*]**

dvw-name-The 1- to 18-character name of the dataview.

VERSION ver-The version of the dataview. The specified version must be in test or production status. If you specify *dvw-name* and omit the version clause, the default version specified in the SET VERSION command is used. Valid versions are:

Tnnn-The version number of a test dataview modeled in the dictionary (CA Datacom/DB native command access and modeled sequential files only).

T-Represents test status.

nnn-The one- to three-digit version number assigned to a production dataview or an unmodeled dataview in test or production status.

PROD-The production status version of the dataview.

Note: You cannot catalog a version in history status.

[WITH] VALIDATION (VSAM only)

Specify this option to verify CA Ideal dataview attributes against the actual characteristics of the VSAM file in the VSAM Access Control Blocks. For more information about validating VSAM dataviews, see section, Runtime Considerations for VSAM Dataviews, in the chapter Creating Dataviews for VSAM Files. If any discrepancies are found, the CATALOG command fails and error messages are issued. You can abbreviate this option as VAL.

Important! Use this option only if the file is accessible in the current environment.

[WITH] NOVALIDATION (VSAM only)

Specify this option to catalog the dataview without validating the attributes. You can specify this option as NOVAL. Use this option if the file is not accessible in the current environment.

WITH NOVALIDATION is the default, but you can change the default by specifying the command SET CATALOG VALIDATION.

When the CATALOG DATAVIEW command is processed, a DISPLAY DATAVIEW command is automatically invoked.

For SQL Access

No CA Ideal components are necessary to create and edit dataview definitions for SQL objects.

For DB2, the definitions of tables and views are maintained in the DB2 catalog. Definitions of dataviews are maintained in the dictionary.

For CA Datacom SQL access, the definitions of tables, views, synonyms, and dataviews are maintained in the Datadictionary of the Datadictionary. Any field information provided in the dictionary, such as headings, initial values, and edit patterns, is incorporated into the dataview object. CATALOG process makes a dataview definition available to CA Ideal.

When you catalog an SQL dataview, CA Ideal records a dataview corresponding to the SQL object as an entity occurrence in the dictionary. The dataview entity occurrence is recorded in the dictionary facility as version 1 in production status. You can issue one or many CATALOG commands in a batch run of CA Ideal.

Deleting an SQL dataview deletes the dataview entity occurrence from the dictionary, and deletes its object member.

Note: The CA Ideal CATALOG command does not update the definition of the SQL object in Datadictionary or the DB2 catalog. Rather, it updates the dataview definition for the SQL object in the CA Ideal dictionary facility.

Once an SQL dataview is cataloged, you can display or print the definition online, delete the dataview entity occurrence from the dictionary facility, or dequeue the dataview if it is improperly enqueued. For more information about how to use dataview, see the *Creating Programs Guide*. For more information about syntax of the DISPLAY, PRINT, DELETE, CATALOG, and DEQUEUE commands, see the *Command Reference Guide*.

Note: SQL access dataviews can be updated by default in CA Ideal.

If the SQL object is an SQL view, the CREATE VIEW statement displays in the dataview Text area.

Readonly Classification

When a dataview for SQL access is cataloged, CA Ideal classifies it as read-only when it refers to a view created with DISTINCT, GROUP BY/HAVING, or a join. A column in a table marked as UPDATE=NO in the SYSCOLUMNS table of the DB2 catalog or created in a view with an SQL function results in a read-only column in the dataview.

For a Sequential File

If a dataview is edited after it was cataloged, you must re-catalog the dataview and recompile any programs that use the dataview as a resource.

For VSAM Files

Before you can use a defined VSAM dataview in a CA Ideal application, the dataview must be cataloged. If a dataview is edited after it was cataloged, you must recatalog the dataview and recompile any programs that use the dataview as a resource.

The CATALOG command for VSAM dataviews lets you specify whether you want to validate the following dataview attributes against the actual characteristics of the VSAM file:

- File organization (ESDS, KSDS, or RRDS)
- Maximum record length (determined by totaling the lengths of the fields defined in the Field Definition fill-in)
- Displacement and length of each primary and alternate key defined in the Dataview Key Definition fill-in

To validate the dataview attributes against the VSAM file, the base cluster and any paths in the upgrade set must be available in the current environment. If these cluster components exist in the current environment, validate the dataview attributes when you catalog the dataview definition. However, if these components do not exist in the current environment, catalog the dataview definition without validating the dataview attributes (the dataview is validated when it is accessed).

Displaying the Dataview Definition

To display a dataview definition, use the DISPLAY command or select option 1 on the Dataview Menu.

The following list contains information specific to each type of dataview.

An CA Datacom/DB native or SQL dataview must be cataloged to CA Ideal before displaying.

- For an unmodeled sequential or VSAM dataview, the catalog listing displays if the dataview definition was cataloged, otherwise the Field Definition displays.
- For unmodeled sequential or VSAM dataviews, the Field Definition fill-in displays for browsing (to edit the dataview, use the EDIT command). You can then display the Identification, Parameter, and Keys fill-in panels by using the appropriate commands or PF keys.

You can display other dataview definition components by explicitly requesting them.

This command has the following format:

```
        {*                       }
DISPLAY {DATAVIEW dvw-identifier } [component]
```

* Represents the current dataview and can be substituted for the dataview identification phrase.

dvw-identifier

The name that identifies the dataview. The *dvw-identifier* includes different parts, depending on the type of dataview to identify:

For SQL dataviews-Specify the dvw-identifier as:

authid.dvw-name

authid (SQL only)**-**The one- to eight-character authorization ID required for SQL dataviews only.

dvw-name-The 1- to 18-character name of the dataview.

For CA Datacom/DB native access, sequential, and VSAM dataviews-Specify the dvw-identifier as:

dvw-name [VERSION ver]

dvw-name-The 1- to 18-character name of the dataview.

ver-The version of the dataview. The specified version must be in test or production status. If you specify *dvw-name* and omit the version clause, the default version specified in the SET VERSION command is used. Valid versions are:

Tnnn-The version number of a test dataview modeled in the dictionary (CA Datacom/DB native command access and modeled sequential files only).

T-Represents test status.

PROD-The production status version of the dataview.

nnn-The one- to three-digit version number assigned to a production dataview or an unmodeled dataview in test or production status.

Note: Unmodeled dataviews created before CA Ideal r2.2 are recognized as version 1, PROD status.

component

For unmodeled dataviews (sequential and VSAM files) only, one of the following or the equivalent PF key:

IDENTIFICATION (PF6)-Displays the dataview identification definition fill-in.

FIELD (PF5)-Displays the dataview field definition fill-in.

PARAMETER (PF4)-Displays the dataview parameter definition fill-in.

KEY (for VSAM dataviews only)-Displays the key definition fill-in.

Dataview Display

The dataview display is slightly different for each type of dataview. Each display starts with a Dataview Message Line, which indicates the type of file the dataview references and shows the date and time when the dataview was cataloged. After the Dataview Message Line, the field display begins with the dataview name shown as the level 1 name. Areas for special information that varies according to file type follow the field display. The following examples show dataview displays for different types of files.

For SQL Dataviews

The display output format for SQL dataviews is shown below. The sample is a dataview for a DB2 view. Notice the ROW ID and TEXT areas following the field display.

```
=>
-------------------------------------------------------------------------------
--
IDEAL: DISPLAY DATAVIEW      DVW IDLSQLM.CUSTADDR            SYS: DOC   DISPLAY
Commnd Seq Level Field name            T I Ch/Dg       K Value
====== =============================== T O P ================================
000001 CATALOGED 11/18/94 14:35       DB2
000002   1 1      IDLSQLM.CUSTADDR
000003   2  2      CUST_ID             X       5       K
000004   3  2      NAME                X      30
000005   4  2      ADDRESS             X      30
000006   5  2      CITY                X      15
000007   6  2      STATE               X       2
000008   7  2      ZIP                 X       9
000009   8  2      PHONE               X      10
000010
000011 ROW ID(s):
000012   IDLSQLM.CUSTIX: CUST_ID
000013
000014 TEXT:
000015 CREATE VIEW IDLSQLM.CUSTADDR
000016              (CUST_ID,
000017               NAME,
000018               ADDRESS,
000019               CITY,
000020               STATE,
000021               ZIP,
000022               PHONE)
000023     AS SELECT
000024               CUST_ID,
000025               NAME,
000026               ADDRESS,
000027               CITY,
000028               STATE,
000029               ZIP,
000030               PHONE
000031     FROM IDLSQLM.CUSTOMERS
====== ============================ B O T T O M =============================
```

For Native Access Dataviews

The following screen shows the dataview display for an CA Datacom/DB native access dataview.

```
=>
------------------------------------------------------------------------------
IDEAL: DISPLAY DATAVIEW      DVW EMPLOYEE (001) PROD         SYS: DOC   DISPLAY
Commnd Seq Level Field name           T I Ch/Dg Occur K Value/Redef/Dep on
====== ================================ T O P =================================
000001 CATALOGED 02/10/94 10:12      DATACOM/DB UPD=YES DBID=001
000002   1  1     EMPLOYEE
000003   2  2     NUMBER              U Z     5       K
000004   3  2     NAME                X      24
000005   4  2     STREET-ADDRESS      X      24
000006   5  2     CITY-ADDRESS        X      15
000007   6  2     STATE-ADDRESS       X       2
000008   7  2     ZIP-CODE-LOC        X       5       K
000009   8  2     SOCIAL-SECURITY     N P     9
====== ============================= B O T T O M ==============================
```

For VSAM Dataviews

The display output for VSAM dataviews is shown below. Notice the different dataview message at the top of the display and the key definitions shown at the bottom.

```
=>
---- ---------  -------------------------------  ----------------------------
IDEAL: DISPLAY DATAVIEW      DVW EMPMASTER (001) PROD        SYS: DOC   DISPLAY
Commnd Seq Level Field name           T I Ch/Dg Occur K Value/Redef/Dep on
====== =============================== T O P ===============================
000001 CATALOGED 04/13/94 09:49       VSAM  KSDS UPD=YES FILENAME EMPMAST
000002                                VAR LENGTH RECORD    MAX RECSIZE  0596
000003   1 1     EMPMASTER
000004   2  2     NAME                                K
000005   3   3     LAST-NAME          X      20       P
000006   4   3     FIRST-NAME         X      15
000007   5   3     MID-INIT           X       1
000008   6  2     EMP-ID              N Z     7       K
000009   7  2     ADDRESS
000010   8   3     STREET             X      20
000011   9   3     CITY               X      15
                                        .
                                        .
                                        .
000021  19  2     DEPENDENTS
000022  20   3     NUM-DEP            N Z     2
000023  21   3     DEP-DATA                       12   DEP ON NUM-DEP
000024  22    4     DEP-NAME
000025  23     5     GIVEN            X      12
000026  24     5     MIDDLE           X       1
                                        .
                                        .

                                        .
000032                           KEY SECTION
000033 Seq File/Path  Field Name                       Unique  Upgrade Set
000034 --- ---------  ------------------------------   ------  ------------
000035  28 EMPMAST    NAME                              PRIMARY
000036  29 EMPID      EMP-ID                           Y       Y
====== ============================= B O T T O M ============================
```

DATAVIEW Message Line

The Dataview Message Line is the first line (or two) in a dataview definition display. This message contains information about when the dataview was cataloged, the type of file the dataview references, and whether the file is updateable. The Dataview Message Line has a slightly different format for each type of dataview. In all cases, the date and time when the dataview was cataloged is indicated by *mm/dd/yy hh:mm*.

For SQL Dataviews

The format of the Dataview Message Line for SQL dataviews is:

```
CATALOGED mm/dd/yy hh:mm       database-type
```

The second line shows the dataview name (the level 1 identifier).

For Native Command Access

For CA Datacom/DB native access dataviews, the format of the Dataview Message Line is:

For z/OS

```
CATALOGED mm/dd/yy hh:mm      DATACOM/DB    UPD=xxx   DBID=xxx
```

For VSE

```
CATALOGED mm/dd/yy hh:mm      DATACOM/DB    UPD=xxx   DBID=xxx
          DEVICE=             MAX-REC-SIZE=    MAX-BLK-SIZE=
```

For Sequential Files

For sequential dataviews, the format of the Dataview Message Line is the same for both modeled and unmodeled dataviews since, after the dataview is cataloged, there is no real difference in the object of modeled and unmodeled dataviews.

For z/OS

```
CATALOGED mm/dd/yy hh:mm      SEQUENTIAL UPD=xxx FILENAME xxxxxxx
                              RECSIZE=
```

For VSE

```
CATALOGED mm/dd/yy hh:mm      SEQUENTIAL UPD=xx FILENAME xxxxxxx
          DEVICE=             MAX-REC-SIZE=       MAX-BLK-SIZE=
```

For VSAM Files

For VSAM dataviews, the Dataview Message Line shows the type of VSAM file, the maximum record size, and the name of the file. The format of the Dataview Message Line for VSAM files is:

For Fixed-length Files

```
CATALOGED mm/dd/yy hh:mm   VSAM  type UPD=xxx FILENAME xxxxxxx
                           FIXED LENGTH RECORD      RECSIZE  xxxx
```

For Variable-occurrence Files

```
CATALOGED mm/dd/yy hh:mm     VSAM  type UPD=xxx FILENAME xxxxxxx
                             VAR OCCURRENCE RECORD    MAX RECSIZE  xxxx
```

For Variable-segment Files

```
CATALOGED mm/dd/yy hh:mm     VSAM  type UPD=xxx FILENAME xxxxxxx
                             VAR SEGMENT RECORD   MAX RECSIZE  xxxx
```

Note: For RRDS files defined with variable-segment records, the second line of the dataview display shows only the record size since RRDS records are always written as fixed-length records.

Dataview Field Display

The remaining components of the formatted dataview definition display are as follows:

- **Command** An area where you can specify line commands. Line commands can copy, move, or delete lines. For more information about description of all line commands, see the *Command Reference Guide*.
- **Seq** A sequence number assigned to each name on the dataview display for reference.
- **Level** The level number that ranks columns in hierarchical order, from 2 through 16. The dataview name is always assigned as level 1.
- **Field name** The name of an elementary (simple) or group (compound) field. If this is a dataview for SQL access, this is the name of a column. A semicolon (;) as the last character indicates that the name is continued on the next line.
- **T (type)** The field or column type. Possible field types are:
  - **X** Alphanumeric (character). Valid for all dataview types.
  - **N** Numeric (signed). For unmodeled files (VSAM and sequential) and CA Datacom/DB native access dataviews, numeric fields can be zoned, packed, or binary. For SQL access, numeric fields can be integer, small integer, and decimal.

- **U** Unsigned numeric. For unmodeled files (VSAM and sequential) and CA Datacom/DB native access dataviews, unsigned numeric fields can be zoned or packed. For SQL access, unsigned numeric fields can be integer, small integer, and decimal.
- **D** Date. For unmodeled files (VSAM and sequential) and CA Datacom/DB native access dataviews, date fields can be zoned, packed, or binary. For SQL access, this form of date field not valid, see the SQL DATE, TIME and TIMESTAMP columns.
- **C** A condition name assigned to a specific value. This is not valid for SQL access dataviews.
- **V** Variable length (variable string and long variable string). This is valid only for SQL access and CA Datacom/DB native access dataviews. For CA Datacom/DB native access to SQL-defined objects, variable length fields are supported, but they display as an X type field and the length shown is the maximum length.

  For each variable length item in an SQL dataview, a two-byte long field for the length is generated in front of the data. This field is not shown in the display. The programmer needs to account for this field only in non-CA Ideal subprograms. Refer to the CALL statement in the *Command Reference Guide*.
- **I** (Internal Type) The internal representation of numeric (signed and unsigned) and date type fields:
  - **P** Packed
  - **Z** Zoned
  - **B** Binary

  Numeric fields can be internally represented in packed decimal, zoned decimal, or binary formats. CA Ideal can operate on most of these directly, performing internal conversions when necessary (floating point formats are not supported). Numeric fields are viewed in terms of decimal (base 10) digits and are fully defined by the total number of integer and decimal places, regardless of internal storage formats. CA Ideal performs all necessary internal conversions on both input and output.

  The CA Ideal format for DB2 numeric columns is shown under the CH/DG entry.
- **Ch/Dg** (Characters/Digits) The length of the field:
  - **Characters-**The number of alphanumeric characters in the field.
  - **Digits-**The number of integer and decimal places (base 10 equivalent) in a numeric field, separated by a period.

  The following example shows the type, internal type, and ChDg values for 5 fields.

Type (Type)	I (Internal Type)	Ch/Dg (Characters/digits)
X		42
X		16

N	P	7
N	Z	10.3
D	P	5 (not valid for SQL access)

In the above example, 42 in the Ch/Dg column for the first alphanumeric field (Type X) indicates a length of 42 characters; and 16 for the next alphanumeric field indicates a length of 16 characters. A numeric field (Type N) with 7 specified in the Ch/Dg column indicates a seven-digit field able to hold seven integer places. Since the internal type is P (packed), the actual size of the field is four bytes ((7 1)/2 = 4). The next numeric field, with 10.3 specified in the Ch/Dg column, indicates a 13-digit field able to hold ten integer places and three decimal places. Since the internal type is Z (zoned), the actual size of the field is 13 bytes. A date field (Type D) with 5 specified in the Ch/Dg column can hold an internal 5-integer date value of up to 273 years from the base year.

SQL numeric fields display as follows:

	T (Type)	**I (Internal Type)**	**Ch/Dg (Characters/digits)**
INTEGER fields	N	B	9
SMALLINT fields	N	B	4
DECIMAL fields	N	P	actual length
NUMERIC fields	N	Z	actual length

SQL date and time fields display with a comment and the following attributes:

	T (Type)	**Ch/Dg (Characters/digits)**
DATE fields	X	10
TIME fields	X	8
TIMESTAMP fields	X	26

A field with an unsupported data type displays as type X and a warning message is issued.

- **Occur** (Number of occurrences)**-**Not present for SQL access dataviews. The number of times a group or field repeats.

  If a number displays in this column (and there is no DEP ON clause in the Value/Redef/Dep on column), it represents the number of times an item (group or elementary field) that repeats a fixed number of times, occurs.

  An item (group or elementary field) can also occur a variable number of times. In this case, the Occur column value represents the maximum number of occurrences, and a DEP ON clause appears in the Value/Redef/Dep on column. The DEP ON (Depending on) field name is specified in the Value/Redef/Dep on column with the clause DEP ON field-name.

  The Depending on field is a non-repeating elementary numeric field with no decimal positions that is located in this dataview preceding the DEP ON clause. At runtime, the Depending on field specifies the actual number of occurrences at any point during the run.

- **K (Key)-**(CA Datacom/DB native access and SQL access) The key column that indicates whether a field is defined as a key or part of a key in the dataview.

  - The field is a full key
  - The field is a partial key (high order portion of a key)

  If the field is not a full or partial key, there is no entry in this column.

  For example, in a dataview where KEY1 consists of FIELD1 and KEY2 consists of FIELD2,FIELD3, the corresponding dataview display identifies the three fields as follows:

  ```
  FIELD1  K  (full key)
  FIELD2  P  (partial key)
  FIELD3   (no entry)
  ```

  Since only the high order portion of a multi-field key can be a partial key, FIELD3 is not a partial key.

  If a key has 65 or more FIELDs defined in the dictionary, CA Ideal only recognizes the first 64 FIELDs, and lists only the first as a partial key.

- **K (Key)** (VSAM only)-Indicates whether the field is a primary or alternate key. If the field is not a primary or alternate key, there is no value in this column:

  - **K** The field defined as a primary or alternate key.
  - **P** The first field in a group that is defined as a key.

  All of the keys, primary and alternate, are listed at the end of the field layout.

- **Value/Redef/Dep on** (Value, Redefinition, or Depending on column-name)**-**An initial value for a condition name or for an elementary field, a field redefinition clause, a DEP ON clause, or a comment.
    - **Value-**An initial default value assigned to this field each time a new record is created. The DVW Administrator specifies it during dataview definition. If no initial value is specified, CA Ideal uses spaces for alphanumeric, 0 for numeric. The default value is a numeric or alphanumeric literal, depending on the type of elementary field. Alphanumeric literals are enclosed in delimiters (“or”).

  Note: For read-only (R) dataviews (UPDATE-INTENT=N), only condition name (Type C) values display since initial value has no meaning for a read-only dataview.

    - **Redefinition-**A Redefinition clause in this field indicates that the field or group is another view of a previously defined field. The field-name is a previously defined field or group that is redefined by this item. This does not apply to SQL dataviews.
    - **Depending on field**-**name-**A Depending on clause in this column indicates that the field or group repeats a variable number of times. The field named in the Depending on clause:
        - Specifies the number of times the field named in the Field Name column occurs in the current record, and
        - Must be a non-repeating, elementary, numeric field that appears previously in the dataview and contains an integer value.
- **WITH IND** (For SQL and CA Datacom/DB Native Access)**-**The clause WITH IND in this column indicates that this column is nullable; that is, it can receive null values.

  Note: For each nullable item, CA Ideal generates a two-byte indicator variable. This variable is not shown in the display. The programmer needs to account for this field only in non-CA Ideal subprograms.
- **:DATE or :TIME or :TIMESTAMP-**For SQL and CA Datacom/DB native access. An indicator of an SQL date, time, or timestamp column that has been converted to an alphanumeric item by CA Ideal.
- **Comments-**For unmodeled sequential and VSAM files, comments provide useful information about the field. A comment is indicated in this column by a preceding colon (:).

- **KEY SECTION-**For VSAM files only, lists the key definitions entered on the Key Definition fill-in:
  - **File/Path-**The file name of the base cluster for the primary key or of the path for any alternate keys.
  - **Field Name-**The name of the key field.
  - **Unique-**For the primary key, the word PRIMARY displays. For alternate keys, a Y indicates that the key is a unique key; N indicates that it is not unique.
  - **Upgrade Set-**For the primary key, this column is blank. For alternate keys, a Y indicates that this path is part of the upgrade set; N indicates that it is not.
- **Row IDs** (SQL access only)**-**The names of any unique indexes (composed of up to 64 non-nullable columns) defined for the underlying table or view for the dataview, and the names of the columns comprising each index.
- **Text** (SQL access only)**-**The text of the SQL CREATE VIEW statement, used if the dataview is associated with a view. If the dataview is for an CA Datacom SQL access synonym, the Text area indicates the object for which this is a synonym.

Printing a Dataview Definition

Use the PRINT command or the PRINT prompter to print a specific dataview definition. To display the PRINT prompter (as it applies to dataview definitions), select option 7 on the Dataview Menu. The following screen contains an example of the PRINT prompt screen.

```
=>
IDEAL: PRINT OCCURRENCE      DVW                            SYS: DOC  PROMPTER

PRINT   DVW ________________________________    VERSION   PROD
        (1)                 (2)                            (3)

  DEST  LIB _____________________________________________________________
        (4)                           (5)

  COPIES  1  NAME ________  DISP REL   DESC ________________________________
        (6)         (7)          (8)                        (9)
-----------------------------------------------------------------------------
(1) DVW = Dataview   PNL = Panel    (2) Name  (3) nnn    (4) Destination:
    MEM = Member     RPT = Report                 PROD       LIB=Output library

    OUT = Ouput      SYS = System                 LAST       SYS=System printer

    PGM = Program    USR = User                              NET=Network printer

    PLA = Plan                                               MAI=Mail

(5) Destination name      (6) # Copies    (7) Name of output

(8) Disposition: REL = Release, KEE = Keep, HOL = Hold

(9) Description (must be in quotes!)
```

The PRINT DATAVIEW command has the following format:

```
      {    *                 }
PRINT {DATAVIEW dvw-identifier }
      {                      }

      [            {MAIL 'email-id'            }]
      [DESTINATION {LIB                        }]
      [            { {SYS }                    }]
      [            { {NET } dest-id [COPIES nn]}]

      [NAME print-output]

      [            {KEEP    }]
      [DISPOSITION {RELEASE }]
      [            {HOLD    }]

      [DESCRIPTION 'string']
```

***** Represents the current dataview and can be substituted for the dataview identification phrase.

dvw-identifier

Identifies the name of the dataview. The *dvw-identifier* includes different parts, depending on the type of dataview to identify:

For SQL dataviews Specify the *dvw*-identifier as:

authid.dvw-name

authid (SQL only)**-**The one- to eight-character authorization ID required for SQL dataviews only.

dvw-nam-The 1- to 18-character name of the dataview.

For CA Datacom/DB native **access, sequential, and VSAM dataviews** Specify the *dvw-identifier* as:

dvw-name [VERSION ver]

dvw-name-The 1- to 18-character name of the dataview.

ver-The version of the dataview. The specified version must be in test or production status. If you specify dvw-name and you omit the version clause, the default version specified in the SET VERSION command is used. Valid versions are:

Tnnn-The version number of a test dataview modeled in the dictionary (CA Datacom/DB native command access and modeled sequential files only).

T-Represents test status

nnn-The one- to three-digit version number assigned to a production dataview or an unmodeled dataview in test or production status.

PROD-The production status version of the dataview.

Note: Unmodeled dataviews created before CA Ideal r2.2 are recognized as version 1, PROD status.

DESTINATION clause

Specifies the output destination.

MAIL 'email-id'-A delimited 1 to 60 character name of a CA- eMail destination.

LIBRARY-The output library.

SYSTEM dest-id-A system printer name.

NETWORK dest-id-A network printer.

COPIES nn

Specifies the number of copies to print on a system or network printer.

Note: This clause is ignored in batch. For a VSE system printer, this clause is ignored both online and in batch.

NAME print-output

Specifies the name assigned to the output if the destination is to the output library.

DISPOSITION

When the destination of an online print request is a system or network printer, the output is placed in the output library, and the DISPOSITION clause has the following effects:

KEEP-The job prints and a copy of the output is retained in the output library.

RELEASE-The job prints and no copy of the output is retained in the output library.

HOLD-The output is held until released.

When a print request is issued in batch and the destination is a system printer, the output is printed on the system printer and no copy is retained in the output library.

When a print request is issued (online or in batch) and the destination is the output library, output is placed in the output library for browsing at the terminal.

For more information about output disposition, see the *Working in the Environment Guide*.

DESCRIPTION 'string'

A 1- to 32-character description of the output, enclosed in delimiters. For a description of valid string delimiters, see the *Command Reference Guide*.

For more information about CA Ideal PRINT command, see the *Command Reference Guide*.

Displaying or Printing an Index of Dataview Definitions

The DISPLAY INDEX and PRINT INDEX commands (options 6 and 7 on the Dataview Menu) list the name and status of each dataview currently in the dictionary facility.

Optionally, the index can include occurrences of entity-types that are related to a dataview (for example, each program related to a given dataview). This display is based on Datadictionary relationships. Refer to the *Command Reference Guide* for the syntax and a description of the command.

The display resulting from the DISPLAY INDEX DATAVIEW command follows:

```
=>
------------------------------------------------------------------------------
IDEAL: DISPLAY INDEX        DVW                              SYS: CCF   DISPLAY

Command Name             Ver S U Type Description             Created  Updated
====== =============================== T O P =================================
000001 APPLSYSTEM        001 H Y                              02/21/94 03/14/94
000002 APPLSYSTEM        002 H Y                              02/21/94 03/14/94

000003 APPLSYSTEM-RO     001 H N                              02/21/94 03/14/94

000004 APPLSYSTEM-RO     002 H N                              02/21/94 03/14/94
000005 @I$IMAGE-BYFILE 001 P Y QSAM Sample program standard   06/10/94 06/10/94
000006 ACCOUNTS          001 P Y                              12/11/94 12/11/94
000007 ACT-DVW-A01       001 P N      SAMPLE DATAVIEW FOR FILE 10/10/94 10/10/94
000008 CMF-VSEIN         001 P N SAM  SYSIN file for CMF input 05/03/94 05/03/94
000009 CMF.APPLSYSTEM          Y DSQL                         09/07/93 12/16/94
000010 CMF.APPLSYSTEM_V
000011                         Y DSQL                         09/24/93 09/14/94
000012 CMFDB2.APPLSYSTEM_V
000013                         Y DB2                          04/27/94 10/22/94
000014 MFSDGR            001 P N VSAM MFS test downgrade      01/04/93 01/04/93
000015 MVS               001 P Y QSAM                         06/15/92 05/20/94
000016 OCC               001 P Y QSAM                         04/13/94 04/13/94
000017 ORDERS            001 P Y                              12/11/93 04/29/94
000018 QAXDVW1V          001 P N VSAM vsam for source transport 07/05/92 12/05/93
```

Command

Specifies an area where you can specify line commands. For more information about description of line commands, see the *Command Reference Guide*.

- **Name** The 1- to 18-character name of the dataview definition. For an SQL access dataview, the name is prefixed by the one- to eight-character authorization ID, as shown on lines 9-13 of the previous figure.
- **Ver (Version)** The one- to three-digit version number CA Ideal assigned when this version of the dataview definition was created.
- **S (Status)** The status of each dataview definition. Status is not used for SQL dataviews.
  - **H** History status
  - **P** Production status
  - **T** Test status

All history entries display first, followed by production versions, then test versions. In each status group, the dataview definition names appear in alphabetical sequence. For each name, versions appear in numerical sequence.

U (Update-intent)

Indicates whether the PDL statements in a CA Ideal application can update records or rows viewed through this dataview. This reflects the table or view definition only, not the individual's authorization for them.

- **Y** The records or rows viewed through this dataview can be updated, inserted, or deleted. An SQL access dataview is always updateable.
- **N** The records or rows viewed through this dataview cannot be updated, inserted, or deleted.

Type

Specifies the type of file represented by the dataview:

- **QSAM-**For sequential QSAM files.
- **SAM-**For sequential SAM files.
- **VSAM-**For VSAM files.
- **DSQL-**For CA Datacom SQL access objects.
- **DB2-**For DB2 objects.
- **spaces-**For CA Datacom/DB native access tables.

Description

Specifies the description for the dataview definition. If the description of an CA Datacom/DB native access dataview is longer than 24 characters, it is truncated on this display.

Created (Creation date)

Indicates the date on which the dataview definition was created or first cataloged.

Updated (Modified date)

Indicates the date on which the dataview definition was last modified or last cataloged.

Editing an Unmodeled Dataview Definition

There are three ways to display an unmodeled dataview definition for editing:

- Enter the following command on the command line.

  ```
  EDIT DATAVIEW dvw-name VERSION version
  ```

If you do not include a dataview name, a prompter displays. Enter the dataview name on the prompter to display the dataview field definitions. You must also enter the version of the dataview to edit. For valid versions, see the section, Using Version Clauses, in CA Ideal Commands in the *Command Reference Guide*.

- Select option 1, DISPLAY/EDIT, from the Dataview Maintenance menu. This displays a prompter. You need to specify not only the name of the dataview, but also whether you want to display or edit the dataview.
- From the DISPLAY INDEX DATAVIEW line command field, enter the line command EDI next to the name of the dataview you want to edit.

Note: A dataview definition must be in test status to edit.

When you edit a dataview definition, the Field Definition fill-in displays by default. To display the Identification fill-in, the Parameter fill-in or the Key fill-in (for VSAM files) immediately, you can include a keyword on the EDIT command entered from the command line. (See the *Command Reference Guide* for information about the keywords for the EDIT DATAVIEW command.)

Deleting a Dataview Definition

There are three ways to delete an unmodeled dataview definition:

- Enter the following command from the command line.

  ```
  DELETE DATAVIEW dvw-name VERSION version
  ```

  If you do not include the dataview name, a prompter displays. Enter the dataview name on the prompter to delete the dataview definition.

  If you do not include the version, an error message displays.

- Select option 4, DELETE, from the Dataview Maintenance menu. The prompter displays so you can enter the name of the dataview to delete.
- From the DISPLAY INDEX line command area, enter the command DEL next to the name of the dataview you want to delete. You can delete multiple dataviews by entering the line command for each dataview you want to delete.

Note: You cannot delete a dataview definition in production status or one that is a resource of a production status program.

The Delete command removes the identification of the dataview stored in the dictionary and the dataview object module in the VLS library.

Duplicating a Dataview Definition

You can duplicate an unmodeled dataview definition to a new name or a new version. There are two ways to duplicate a dataview definition:

- Enter the following command from the command line:

  ```
  DUPLICATE DVW dvw-name [VERSION version]

       [NEWNAME newname]
  [TO] [NEXT [VERSION] ]
  ```

If you do not include the name of the dataview to duplicate, a prompter displayed. If you do not include the NEWNAME clause, the dataview is duplicated to the next version.

- Select option 5, DUPLICATE, from the Dataview Maintenance menu. The following prompter displays so you can complete the command.

```
=>
----------------------------------- ----------------------------------------
IDEAL: DUPLICATE            DVW                            SYS: DOC  PROMPTER

DUPLICATE DVW __________________  VERSION 001  SYSTEM ___ TO _______ ________
          (1)       (2)                    (3)         (4)      (5)     (6)

------------------------------------------------------------------------------
(1) MEM = Member    RPT = Report    (2) Name         (3) nnn=Ver #
    PGM = Program   SYS = System                           LAST
    PLA = Plan      USR = User                             PROD
    PNL = Panel     DVW = Sequential Dataview

(4) Original System id              (5)-(6) "NEXT" "VERSION"
   (Default current)                                or
                                            "NEWNAME"  name
                                        (For DVW,PGM,PLA,PNL,MEM,RPT)
```

Duplicating to a New Version

When you duplicate a dataview definition to the next version, the entire definition is duplicated and the Field Definition fill-in of the new version displays for editing.

When you duplicate a dataview definition that was created before release 2.2, the original dataview definition is modified to point to a new version 1 with an ACCESS-CODE of P (production). Then version 2 is created from the production dictionary entry and the VLS for the production dataview. (For more information about how ACCESS-CODE determine status for unmodeled dataviews, see the section titled Changing the Status of a Dataview Definition that follows.)

Duplicating to a New Name

When you enter the DUPLICATE command with the NEWNAME option, a new dataview definition is created with the same parameters, fields, and keys (for VSAM files) as the original dataview. The Identification fill-in displays so you can enter the identification information to record in the dictionary.

When you enter any information on the Identification fill-in, you can press the PF5 (Field) key to display the Field Definition fill-in in edit mode. You can make any changes, using any of the function keys, just as you would when creating a dataview definition.

Changing the Status of a Dataview Definition

When you create a dataview definition for an unmodeled file (sequential or VSAM), the definition is in test status. The MARK STATUS command changes the status of an unmodeled dataview definition to production or history status. If you enter an incomplete MARK STATUS command or select option 5 from the Dataview Maintenance Menu, a prompter displays. The prompter indicates the options that are required to complete the command and explains the possible values that you can enter for each option. When you complete the prompter and press the Enter key, the command is passed to CA Ideal for processing.

A definition is marked from test to production status when the decision is made that no further testing is necessary and the definition is ready to use in a production application. Production versions are subject to the following rules:

- There can be only one production version of a dataview definition at a time. You cannot edit or delete that version.
- If the production version of a dataview definition was not named as the resource of a production program, you can mark the dataview definition directly to history.
- Marking another test version of a dataview definition to production automatically retires any existing production version to history status.
- You cannot mark a production version of a dataview definition that was named as a resource of a production program to history without first replacing it with a new production version.

Status for Unmodeled Dataviews

Although dataviews are an extension of the data model, CA Ideal enforces the version and status for unmodeled dataviews, not the dictionary. For unmodeled dataviews, the entity-occurrence is stored in the dictionary in test status, but CA Ideal enforces production and history restrictions based on a value recorded in the ACCESS-CODE field in the dictionary. Thus, if a production and a test version exist for an unmodeled dataview, there are actually two test versions in the dictionary, one with an ACCESS-CODE of PROD and one with an ACCESS-CODE of TEST.

PF Key Assignments

The PF key assignments shown below are in effect while using the dataview definition facility. Commands in bold are assignments consistent throughout all facilities of CA Ideal. For more information about description of all CA Ideal commands, see the *Command Reference Guide*.

Help (PF1)

Displays a panel or series of panels containing information that explains the current function.

Return (PF2)

Returns from a help panel to the dataview component display or from the dataview to the menu that selects the dataview.

Print Screen (PF3)

Generates a hardcopy printout of the current screen contents.

Parameter (PF4)

Positions to the parameter fill-in.

Field (PF5)

Positions to the field description fill-in of the dataview.

Identification (PF6)

Positions to the identification fill-in of the dataview.

Scroll Backward (PF7)

Displays the previous frame in the current component.

Scroll Forward (PF8)

Displays the next frame in the current component.

Find (PF9)

Finds the next occurrence of an alphanumeric literal previously specified in a full FIND command. When specifying a command with range specifications, the specified range specifications are used from the previous command.

Scroll Top (PF10)

Positions to the first line of the component.

Scroll Bottom (PF11)

Positions to the bottom of the component.

Input (PF12)

Opens a window of null lines preceding the first line of the component or at the current cursor position. Unused null lines in the window are deleted when you press the Enter key after INPUT.

Dataview Access Security

You can establish a relationship between a dataview and one or more CA Ideal systems in Datadictionary. This relationship, SYS-DVW-USE, restricts access to the dataview to users on one of the related systems. This means that the dataview cannot display or print unless you are currently on a related system. Also, you cannot include the dataview in a program's resource fill-in unless the program is in a related system.

Any user authorized for any system can access dataviews not related to a system, the default.

For more information about how to define the SYS-DVW-USE relationship, see the Datadictionary documentation.

Chapter 2: Creating Dataviews for SQL Access

When a CA Ideal application uses SQL to access data, it requires a dataview for the SQL object-table, view, or synonym. The dataview includes all fields defined for the SQL object. It is the only dataview that can be defined for it. The dataview name is the name of the SQL object.

You can use SQL dataviews in embedded SQL DML statements. However, for CA Datacom SQL access, you can use DDL and DCL statements, such as CREATE and GRANT, without defining CA Ideal dataviews, which do not have to be placed in the resource table component of the program.

Using CA Ideal PDL, you can find any row or set of rows by using the FOR statement with the dataview for the table, view, or, for CA Datacom SQL access, synonym. See the *Programming Reference Guide*.

No CA Ideal components are necessary to create and edit dataview descriptions for SQL objects. For DB2, the definitions of tables and views are maintained in the DB2 catalog, while definitions of dataviews are maintained in the dictionary. For CA Datacom SQL access, the definitions of tables, views, synonyms, and dataviews are maintained in the dictionary.

The catalog process makes a dataview definition available to CA Ideal. When you catalog a SQL dataview, CA Ideal records a dataview corresponding to the SQL object as an entity occurrence in the dictionary. The dataview entity occurrence is always version 1 in production status.

For more information about cataloging a dataview for SQL access, see Cataloging a Dataview in the “Creating Dataviews for CA Datacom/DB Native Command Access” chapter.

Rules for SQL Dataviews

CA Ideal imposes certain rules on dataviews used for SQL access. The rules are enforced by the CA Ideal CATALOG command. For more information, see the *Command Reference Guide*.

Identifying Columns

- CA Ideal accepts authorization IDs of up to eight bytes for SQL dataviews.
- CA Ideal dataviews accept column identifiers of up to 32 bytes. However, the maximum length depends on the SQL mode. For DB2, the maximum length is 18 characters.
- For the names of columns, CA Ideal uses the character column name.
- The column names in an SQL dataview cannot be delimited identifiers. To reference existing columns named with delimited identifiers, you can create a view of the object containing the columns using valid CA Ideal column names and catalog the view in CA Ideal.

Data Types of Columns

CA Ideal supports the following SQL data types:

SQL Data Type	**Data Type in CA Ideal**
Character	Alphanumeric
Variable string	Variable length
Long variable string	Variable length
Numeric	Zoned numeric (CA Datacom SQL); packed numeric (DB2)
Decimal	Packed numeric
Integer	Binary full word
Small integer	Binary half word
Date	Ten-byte alphanumeric
Time	Eight-byte alphanumeric
Timestamp	26-byte alphanumeric

Date and time take the length defined by a local database management system exit if it is the default. CA Ideal supports the SQL limits on length and precision, except for long variable string columns, which are limited to 16,000 bytes for CA Datacom SQL access or 32,000 bytes for DB2.

Unique Index Requirement

In the following circumstances, dataviews for DB2 and CA Datacom/DB ANSI mode must access a single table (not the result of a join) that has at least one unique index:

- In a FOR statement that has an ORDERED BY clause and is used for updating
- In a FOR FIRST clause used for updating, with a logical scope that includes a PDL TRANSMIT, CHECKPOINT or BACKOUT statement or an SQL COMMIT or ROLLBACK

Reserved Words

You cannot use PDL reserved words as the names of dataviews.

NonSQL Field Information

For CA Datacom SQL access dataviews, field information provided in the dictionary, such as headings, initial value, and edit pattern, is incorporated into the dataview object, regardless of the intended access method.

SQL Plans

SQL plans are needed for any CA Ideal program that names an SQL access dataview as a resource. For CA Datacom SQL access, plan options are part of the program environment data and are entered as part of the program definition. For more information about setting plan options in the program definition, see the *Creating Programs Guide*. For more information about plan options for CA Datacom SQL access and their defaults, see the CA Datacom/DB *Programming with SQL Guide*.

For CA Ideal programs that access DB2 tables and views, the application plan is defined in CA Ideal. The plan is then generated and bound using the CA Ideal command GENERATE PLAN. For more information about defining and generating DB2 plans for CA Ideal applications, see the *Administration Guide*.

Recatalog and Recompile

CA Ideal captures all information about a dataview, including column attributes, when the dataview is cataloged. This information is incorporated into the program when the program is compiled, and actually becomes part of the program object module.

If a change is made that affects the columns comprising a table or view (for example, a view is dropped and recreated with an additional column), then you must recatalog the table or view to CA Ideal and recompile every program using the dataview.

Note: If you change a view that does not affect existing column definitions, you can run the programs using the associated dataviews without recompiling, but the SQL text that defines the view is not correct on the dataview display panel.

Specifying Which Database to Access in a Mixed SQL Site

A site that has both CA Datacom SQL support and DB2 can access objects in both database management systems from the same program. The catalog process records which database a particular dataview accesses. At installation, a default database type is specified and any dataview is assumed to access the default database type. You can set a new default using the command:

```
SET [SITE] ENVIRONMENT SQL dbms
```

The value of *dbms* is either DB or DB2. You can override the default for a particular dataview using an option on the CATALOG command:

```
CATALOG dbms DATAVIEW auth-id.obj-name
```

Chapter 3: Creating Dataviews for CA Datacom/DB Native Command Access

This chapter describes the use of dataview definitions for CA Datacom/DB native command access.

Defining CA Datacom/DB Native Dataviews

The data administrator uses Datadictionary in batch or online to maintain table and dataview definitions. There are no facilities in CA Ideal to describe and maintain descriptions for CA Datacom/DB CBS dataviews since CA Ideal has access to the Datadictionary.

Data Structures

This section describes the data administration functions of the CA Ideal user and assumes some familiarity with the Datadictionary. It explains how to establish and maintain modeled dataview definitions for CA Datacom/DB tables for CA Ideal use. For more information about explanation of these terms and use, see the CA Datacom/DB documentation.

Because dataview definitions are stored in Datadictionary, your database administrator should establish and maintain them using Datadictionary. Step by step instructions on defining a dataview using Datadictionary are provided in the section titled Defining Dataviews with Datadictionary in this chapter. However, to be sure you are creating a valid dataview for CA Ideal, you should familiarize yourself with the rules CA Ideal enforces when cataloging a dataview (See the Cataloging a Dataview section in "Introduction" chapter) before defining the dataview in the dictionary.

Access Datadictionary Online by selecting option 3 on the Administration Maintenance Menu or by issuing the CA Ideal command:

```
DDOL
```

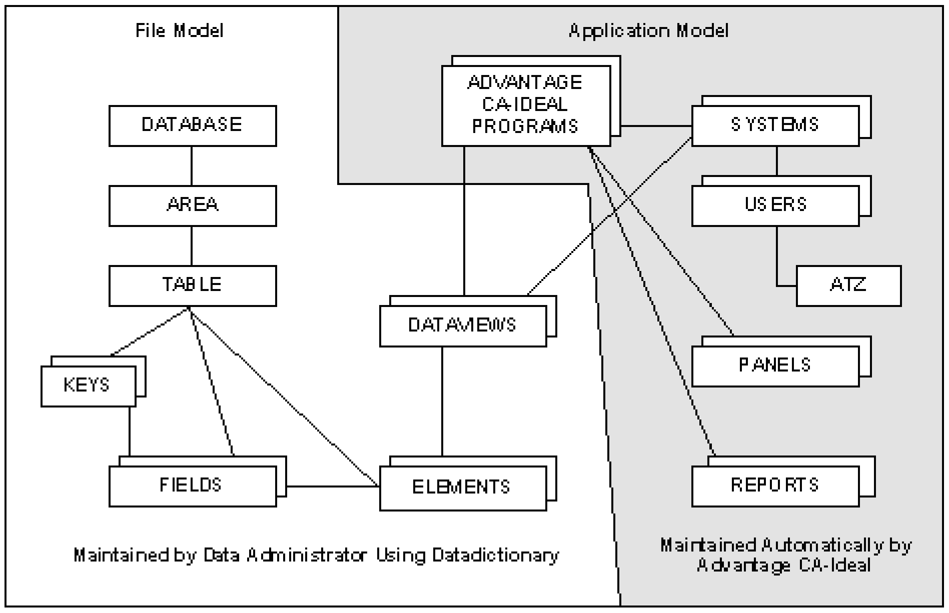
At any time during DDOL use, the Data Administrator can return to the CA Ideal Main Menu by issuing the DDOL command:

```
IDEAL
```

The data administration functions of a Data Administrator include:

- Defining the *table* or *record layout* in the dictionary facility, including establishing fields and keys.
- Defining *elements* of the record in the dictionary facility. An element is a group of one or more contiguous fields in a record.
- Defining *dataview definition* occurrences in the dictionary facility.

The following graphic illustrates the entity types and relationships in CA Datacom/DB and their relationship to CA Ideal. In this graphic, the unshaded area represents the file model, and the shaded area represents the application model.

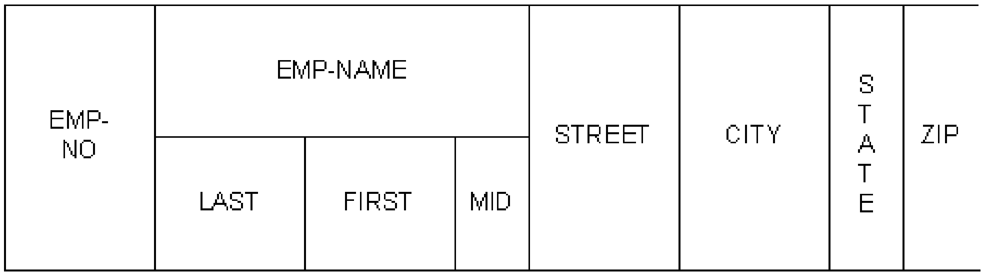

File Model
Application Model
ADVANTAGE CA-IDEAL PROGRAMS
DATABASE
SYSTEMS
AREA
USERS
TABLE
ATZ
DATAVIEWS
KEYS
PANELS
FIELDS
ELEMENTS
REPORTS
Maintained by Data Administrator Using Datadictionary
Maintained Automatically by Advantage CA-Ideal

File Model

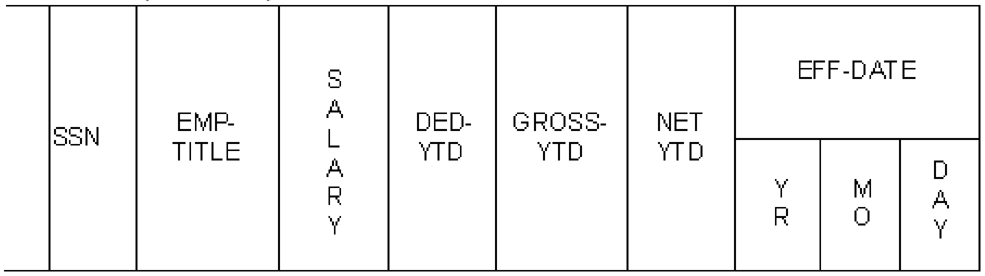
The process of defining the record layout and establishing key fields is fully explained in the CA Datacom/DB documentation. In addition to file model information required by the dictionary facility, the Data Administrator should refer to the CA Datacom/DB documentation for descriptions of utilities for maintaining the dictionary definitions.

For the remainder of this section, assume the following figure is a record layout as defined in the dictionary facility for this file.

PER-REC



PER-REC (continued)



Defining Elements

The definition of elements is determined by the needs of various applications for the data in the table. An element is the logical unit of data transferred by the CA Datacom/DB software.

In CA Datacom/DB, you can define a dataview as any ordered collection of elements, with an element being any collection of fields in a record. This can include the entire record or just one field in the record. An element must be defined as consisting of fields that are contiguous.

The elements related to an CA Datacom/DB native dataview cannot contain overlapping columns. Otherwise, an error occurs when the dataview is cataloged.

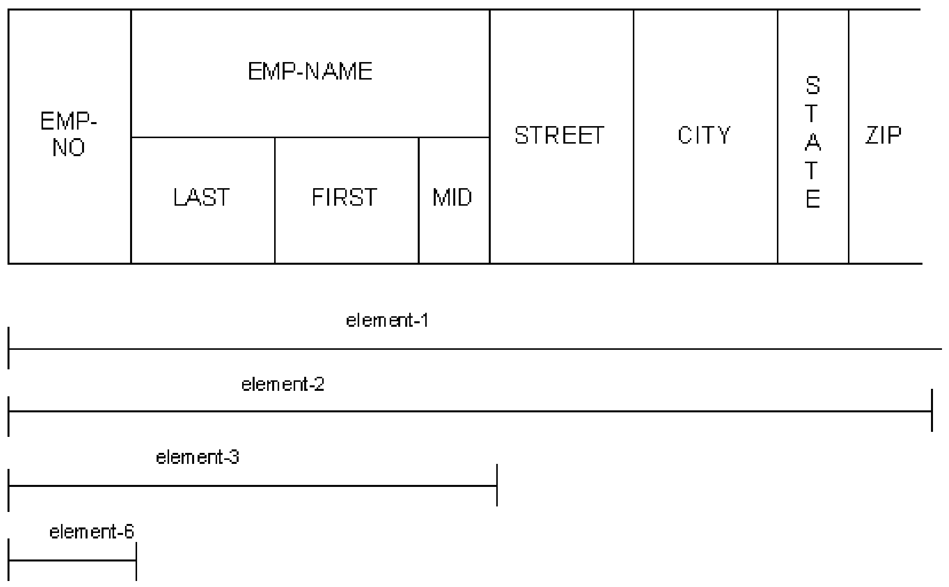
CA Ideal, however, imposes additional restrictions that limit the elements that can be related to a given dataview (for CA Ideal use) and how those elements can be defined.

The following restrictions apply for dataviews and their related elements that CA Ideal accesses:

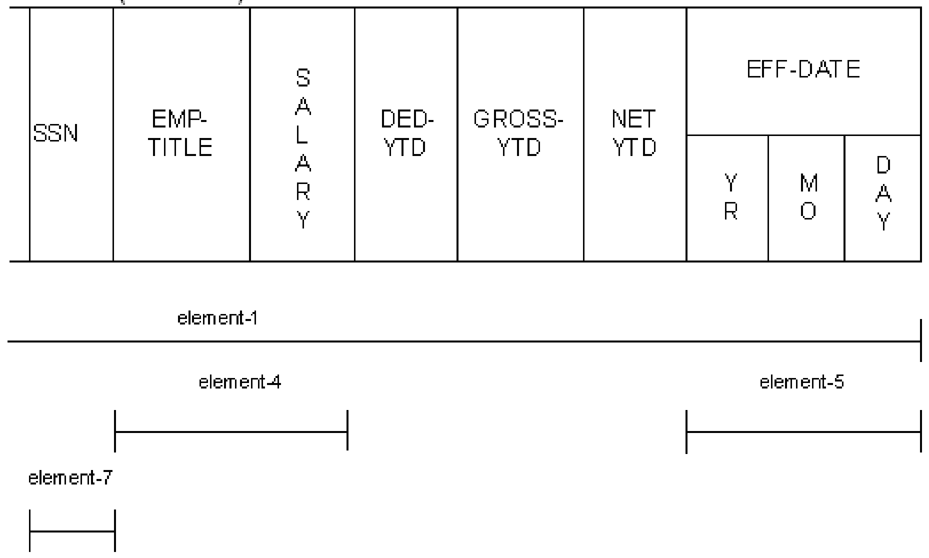
- An element related to a CA Ideal dataview must be explicitly defined as a collection of contiguous fields. In particular, when performing the DEFINE ELEMENT or the MODIFY ELEMENT function in DD, once the starting and ending fields are selected, all intervening fields must be explicitly included in the element. If fields are added to the table (when it is in TEST status), the additions are *not* automatically reflected in the table's related elements. The DA must also execute DEFINE ELEMENT or the MODIFY ELEMENT maintenance and explicitly include the new fields for CA Ideal if they are in the range of an existing element that CA Ideal uses.
- If a compound (group) field is included in an element, all of its subordinate fields must also be explicitly included in the element. If a field is redefined, not all redefinitions need to be explicitly included. However, the redefined field must be explicitly included if any of the redefining fields are included.
- Each dataview must be comprised of elements from a single version of the same record (and, therefore, the same table). However, you can nest the FOR statement in the Procedure Definition Language (PDL) to dynamically join two or more dataviews together.

The following graphic illustrates the definition of elements for a sample record, PER-REC, which could be defined in the dictionary facility.

PER-REC



PER-REC (continued)



Defining Dataviews

The Data Administrator defines a dataview definition as a named and ordered collection of elements. However, as the CA Ideal application developer, you are concerned with only the constituent columns of these elements. More than one dataview can share an element. Any number of applications can share a dataview.

Using the elements defined in the previous graphic, the following are some of the dataviews that could be defined:

```
EMPLOYEE          =       element-1
EMPLOYEE-IDENT    =       element-3
EMP-NO-SSN        =       element-6
                          element-7
EMP-HIST          =       element-2
                          element-4
                          element-5
EMP-GEN           =       element-3
                          element-4
```

Since element-1 and element-2 have overlapping columns, the following example is illegal:

```
EMP-INVALID       =       element-1
                          element-2
```

Rules for CA Datacom/DB Native Command Access

The following rules are enforced when you catalog a dataview for CA Ideal use. You must be aware of these rules when you are defining the dataview in the dictionary since CA Ideal, not Datadictionary, enforces these rules.

Dataview Status and Related Occurrences

A dataview can be successfully cataloged in TEST or PROD status. A dataview that is cataloged in TEST status must have all elements in TEST status. All version numbers of the elements must be the same as the dataview version. A dataview that is cataloged in PROD status must have all elements in PROD status. The elements in PROD status do not have to have the same version number as the dataview.

The database structure for all tables in that database or table structure must be enabled in Datadictionary so that the dataviews can be cataloged in CA Ideal.

Note: Exceptions to the descriptions in this section exist for dataviews created before Datadictionary Release 2.4. Refer to Datadictionary documentation for details.

DATAVIEW Information

For a dataview that CA Ideal accesses, the only DATAVIEW information whose specified value is used is UPDATE-INTENT. You can specify Y (to allow updates) or N (to prevent updates). CA Ideal interprets any other value as N. If the DATABASE entity-occurrence name is DATA-DICT or UNIVERSAL, CA Ideal assumes a value of N (to prevent updates) for UPDATE-INTENT.

CA Ideal uses the DATAVIEW attributes DBMS-USED and ACCESS-METHOD with values it supplies from other sources.

Setting a DBID

When the DATABASE is created, it is defined with a DBID. This DBID is cataloged in the dataview and passed to the database at execution time. If the DBID changes or a table with a matching definition exists in another database, then non-SQL programs can be repointed to the new DBID. The CA Ideal commands ALTER PROGRAM, ASSIGN DATAVIEW, and ASSIGN DBID modify the program object, eliminating the need to recatalog the dataview or recompile the program.

Dataview Key Relationship

Although the file model in the dictionary lets the DA relate dataviews to keys through the relationship DVW-KEY-ACCESS, CA Ideal does not use these relationships. CA Ideal searches for data according to the fields specified in the program FOR construct's WHERE clause. The application developer does not need to know about actual keys. During execution of an application, CA Ideal searches for the data using an optimized technique that takes advantage of any set of actual keys in the associated file. Therefore, when creating dataview definitions for CA Ideal use, the DA does not need to relate the dataviews to keys since CA Ideal ignores these relationships.

Cataloging the Dataview

A dataview must be cataloged before CA Ideal can use it. After creating an CA Datacom/DB native dataview using Datadictionary, use the CATALOG DATAVIEW command in CA Ideal.

Access the CATALOG prompter by selecting option 2 on the CA Ideal Dataview Menu. For more information about syntax and description of command, see the section Cataloging a Dataview.

TABLE Attributes

The TABLE attribute DBMS-USED must be CA Datacom/DB.

The attribute ENTY-HIST-VER automatically deletes history status entities in the dictionary database. You can set the number stored in this attribute differently for any file in the dictionary database. Datadictionary uses this attribute when new versions of a dictionary entity are copied or modified into production status. After a new entity is copied to production, the previous PROD version is automatically copied to History. Datadictionary consults the file entity in the dictionary database and deletes the oldest history status version if the number of history occurrences is greater than the value specified in the ENTY-HIST-VER attribute. The default value for ENTY-HIST-VER is three. You can modify it.

The Datadictionary automatic deletion of history status entities deletes these entities from the dictionary without performing corresponding deletes of VLS members. This attribute affects the maintenance of the following CA Ideal entities in VLS libraries: Program, Panel, Report, and Dataview.

Variable Length Records

All tables defined to CA Datacom/DB are considered fixed-length records. The VSAM/Transparency feature of CA Datacom/DB can simulate variable length records. Records nominally defined to the dictionary as fixed-length can take up different amounts of space per record if data compression is used.

CA Ideal does not support variable length fields in CA Datacom/DB dataviews nor multiple length records of different lengths.

For example, field A can be between 50 and 80 bytes.

CA Ideal supports variable repeating fixed length groups. For VSAMT dataviews, you must define the Datadictionary structure with the following rules:

1. The dataview must consist of one element that contains the entire table definition.
2. The variable occurring group must be at the end of the record. You must define the maximum number of times the group can occur (REPEAT) and identify a field for the DEPENDING ON value on the field attribute definition. There can only be one variable occurring group.
3. The user compression exit in the CXX must be DBVVRPR. This is what tells DB this is a variable length table. The Datadictionary table attribute USER COMPRESSION must be set to DBVVRPR and CA Datacom/DB COMPRESSION set to No.

This is what tells CA Ideal at catalog time that the dataview is variable length. If both the dictionary and the CXX do not contain the exit name, element length discrepancy messages probably result. For example, if CA Ideal does not catalog the dataview as variable, it expects the maximum length. DB returns the actual variable length of the entire record.

You can identify a dataview as a variable length VSAMT dataview. It displays an extra line that indicates this is a VAR OCCURRENCE RECORD for an CA Datacom/DB type dataview.

The following table displays a variable length VSAMT dataview:

```
>------------------------------------------------------------------------
IDEAL:  DISPLAY DATAVIEW     DVW IDEAL-VSAMT(T001)     SYS:KTD


Seq Level Field name      T I  Ch/Dg  Occur  K Value/Redef/Dep on
=========================== T O P =================================
CATALOGED 12/05/94 17:49  DATACOM/DB UPD=YES DBID=052

                          VAR OCCURENCE RECORD
1  1    IDEAL-VSAMT         X      4     2
2  2    SIMPLE-REPEAT       X       4         K
3  2    KEY-FIELD           X       3
4  2    ODO-FIELD           U P     3
5  2    ODO-REPEAT          X       4   14  DEP ON ODO-FIELD
=========================== B O T T O M ===========================
```

Elements of a Dataview

Rules for elements of a dataview are as follows:

- Each CA Datacom/DB native dataview is comprised of elements of the same record. However, you can nest FOR statements in the PDL language to dynamically join two or more dataviews together.
- An element must be defined with explicitly contiguous columns for CA Ideal access. It is possible to violate this rule accidentally by inserting a field in the record through DEFINE TABLE, but forgetting to include the field in the appropriate DEFINE element or MODIFY element.

Data Types of Columns

Rules for data types of columns are as follows:

- CA Ideal supports the following data types:
  - Alphanumeric (display) (C)
  - Zoned decimal (N)
  - Packed decimal (D)
  - Binary (B)
  - Date (D)
- CA Ideal converts all other, unknown data types to alphanumeric and issues a warning message.

- CA Ideal supports numeric columns (or fields) up to the equivalent of 31 decimal (base 10) digits (for zoned or packed decimal) and up to the equivalent of 9 decimal (base 10) digits (full word or half word binary) for binary fields. CA Ideal supports only half word (2 bytes) and full word (4 bytes) binary fields. Binary fields of other than 2 or 4 bytes (including data types 2, 4, and 8) can be defined, but CA Ideal treats them as alphanumeric fields (with a warning message) and the user must call a non-CA Ideal subprogram to handle them. CA Ideal does not support floating point items of any precision.
  - A native dataview can contain null eligible columns.
  - DATE, TIME, and TIMESTAMP columns are supported. When the PDL statement FOR NEW is executed at runtime, the columns with these data types are initialized with the current date or time.

Other Field Attributes

CA Ideal ignores FIELD occurrences with CLASS=I (INDEXED BY). The BLANK-WHEN-ZERO attribute is also ignored. If either the LOW-RANGE or HIGH-RANGE attribute is used with a CLASS=V FIELD, a warning message occurs.

The REDEFINES attribute has the following values:

- **N-**(Does not redefine) is the default
- **P or S-**Treated as P (redefines the previous FIELD occurrence at the same level).

VariablyRepeating Occurrences

The CA Ideal user is restricted to one variably-repeating (DEPENDING-ON) item in a dataview. The DEPENDING-ON control field must be a non-occurring elementary numeric identifier (with zero decimal positions) that appears previously in the same dataview. The variably-repeating field or group and all of its subordinates, if any, must be at the end of the data structure. In other words, a variably-repeating simple (elementary) field must be the last field in a dataview. A variably-repeating compound (group) item can have subordinate items, but it cannot be followed in the dataview by any non-subordinate fields. A variably-repeating group cannot have initial values. Also, a variably-repeating item cannot be embedded in another repeating group.

Number of Levels of Occurrences

CA Ideal supports a maximum of three nested levels of occurring fields (that is, tables of up to three dimensions are supported).

Levels of Data Structure Nesting

The maximum number of nested levels of groups and fields in a dataview is 15. Since the dataview itself is the outermost level (level 1), the highest level number allowed is 16. Since the dictionary assigns level 1 to all columns with a PARENT of START, the level numbers CA Ideal assigns is always at least one greater than the level numbers assigned by the dictionary.

Compound Fields

Compound fields (group fields) should not have initial values. If they do, the value is ignored and a warning message issued.

Edit Patterns

You can specify any valid COBOL edit pattern (except Sterling currency symbols) for a field in the dictionary (attribute EDIT-PATTERN). CA Ideal reports use it unless overridden by RDF. For more information, see the *Generating Reports Guide*. The PDF also uses the edit pattern specified in the dictionary for dynamically generated panel fields.

Field Column Headings

You can specify field column headings for fields in the dictionary (attributes HEADING-1 and HEADING-2). However, headings longer than 20 characters are truncated to 20 characters in the CA Ideal cataloged dataview. PDF uses the field column headings specified in the dictionary for dynamically generated panel fields. They are also used in CA Ideal reports unless overridden by RDF.

Note: CA Ideal uses a not-sign (⍰) as the begin-line character in report headings. For more information, see the *Generating Reports Guide*.

PDL Reserved Words

You cannot use a PDL reserved word as the name of a dataview. See the reserved word list in the *Command* Reference *Guide*. You can use reserved words for field names; however, qualification of such field names with the dataview name is required. For example, assume there is a dataview called SHIPMENT with elements containing fields called TO and FROM. It is valid for a PDL procedure to have statements:

```
MOVE SHIPMENT.TO TO LOCATION
SUBTRACT SHIPMENT.FROM FROM AMOUNT
```

However, the following is ambiguous and causes a program compilation error:

```
MOVE TO TO LOCATION
```

Use of Field Entity Names

For names of fields, CA Ideal uses the 32-character field entity-occurrence name, not the 30-character compiler name. Field names in CA Ideal must be unique in the dataview. This occurs automatically since all elements must belong to one record and Datadictionary requires field names to be unique in the record name.

MultiValue Condition Names

CA Ideal supports one unique value for each condition name (class V field in the dictionary). Lists or ranges of values are not supported.

Figurative Constants

CA Ideal supports the following figurative constants as values in the dictionary:

```
SPACE   SPACES
ZERO    ZEROS     ZEROES
```

The VALUE attribute for a condition-name (class V field in the dictionary) cannot be blank. A value of all spaces can only be represented by the figurative constants SPACE or SPACES.

Optional FIELD Attributes

CA Ideal recognizes the following considerations for optional FIELD attributes:

- **SEMANTIC-TYPE** Entered for date fields only. Enter a SEMANTIC-TYPE of CA-DATE for a numeric field (TYPE=N, D, or B) that contains an internal date stored as the number of days since (positive number) or preceding (negative number) the date December 31, 1900. A five-digit number can contain values from -99,999 through 99,999, or the equivalent of almost 274 years before and after the base date. This would be from approximately the year 1627 to approximately the year 2175. A six-digit number can represent dates approximately 2740 years before and after the base date, and so on. (CA Datacom/DB native access)
  A SEMANTIC-TYPE of SQL-DATE may be present for a field (column) that was created via SQL. CA Ideal will process the binary format held on the database and present the same character string format as would be provided with SQL access to the same data. The same applies to the SQL-TIME and SQL-TIMESTAMP types.
- **NULL-INDICATOR** may be "Y" for fields defined as SQL columns. CA Ideal will handle these fields in the same way for native access as it does for SQL access.
- **DECIMALS** The total number of decimal (fractional) places in the number. CA Ideal determines the number of integer places by calculating the total number of decimal (base 10) digits that fit in the field (based on the TYPE) and subtracting the DECIMALS value.

- **DEPENDING-ON** The name of a non-occurring simple numeric field with DECIMALS=0 precedes this field in the record. The CA Ideal restriction is that a field with a DEP-ON clause can be a simple or compound field, but must not be followed in the dataview by any other fields (unless they are subordinate fields to a compound field).
  - **EDIT-PATTERN** A COBOL-like edit pattern used when displaying or printing the field. The CA Ideal Report Definition Facility and the CA Ideal Panel Definition Facility use this pattern as the default for the field. If you omit this edit pattern, the following default edit pattern is used:
  - **X(n)** Alphanumeric fields (where *n* is the length)
  - **9(i).(9d)** Numeric fields (where *i* = integers, *d* = decimals)
- **HEADING-1 and HEADING-2** One or two literals (without quotes) used as the default column headings for the CA Ideal Report Definition Facility and for dynamic field definition using the CA Ideal Panel Definition Facility. For example, if ACCOUNT is specified as the value for HEADING-1 and NUMBER is specified as the value for HEADING-2, the column header appears as follows:

  ACCOUNT NUMBER
- **SIGN** N to force “unsigned” (the default for numeric fields is Y).
- **VALUE** For numeric fields, a valid numeric literal, or one of: ZERO, ZEROS, ZEROES.

  For alphanumeric fields, enter the characters to appear as the initial value without quotes (or any other delimiter) or enter one of: SPACE or SPACES.

  For a field with CLASS=V (value or condition-name in CA Ideal), you must type at least one non-blank character for VALUE. If the condition-name corresponds to a value of spaces, type the word SPACE or SPACES.

  If a value contains embedded blanks, type in the value with no delimiters. For example, enter a three-character value “A B” as A in position 1, blank in position 2, B in position 3.

CA Ideal does not support any of the following:

- Multiple values for a single condition-name.
- A range of values for a field, either in the VALUE attribute or the LOW-RANGE and HIGH-RANGE attributes.
- COBOL figurative constants:
  - HIGH-VALUE(S)
  - LOW-VALUE(S)
  - QUOTE
  - ALL literal

INCLUDENILKEY

When defining a key in CA Datacom/DB, there is a parameter called INCLUDE-NIL-KEY.

This *nil* value has nothing to do with the *null* value used in DB2 and CA Datacom/DB. In most cases, you should define INCLUDE-NIL-KEY as YES. When this parameter is set to NO, an alphanumeric key value of spaces or a numeric key value of binary zeroes is considered nil and is not included in the index. Although CA Ideal has no control over the processing that takes place when this parameter is used, its improper use can be the culprit of poor performance in CA Ideal applications.

You might assume that you should use INCLUDE-NIL-KEY=NO whenever keys have a high occurrence of the nil value to free up index space. In fact, you should limit the use of this parameter to very specific situations:

- Use INCLUDE-NIL-KEY when it is not necessary to access records that have an alphanumeric key value of spaces or a numeric key value of binary zeroes.
- A two-byte binary key field designated as DAYS-LATE with INCLUDE-NIL-KEY=NO keeps non-delinquent accounts with zero days late (a nil value) from being indexed. If only a small portion of the table contains delinquent accounts, a considerable savings results by eliminating both the cost of indexing and storing index entries for non-delinquent accounts.
- Be aware of the consequences when choosing to use the INCLUDE-NIL-KEY=NO with CA Ideal. If the range of the search condition of a WHERE clause specifies the key field that has INCLUDE-NIL-KEY=NO and that range includes the nil value, then Compound Boolean Selection is forced to do a full file traversal to retrieve the data. You cannot use the key for access because the key cannot access all possible rows that satisfy the request. Consider this example:

  ```
  FOR ACCOUNTS
     WHERE DAYS-LATE LE 0
     .  .  .
  ENDFOR
  ```

  Because the nil value is included in the key range, CBS cannot use this key to access the data records. All records must be read to return the correct data to the program.

  Adding a non-keyed field to the WHERE constructs a temporary index for the same reason-the key cannot be used because it does not point to all possible records that could contain the non-key value.
- If you do define a key as INCLUDE-NIL-KEY=NO, be sure that it reflects the way the data is accessed and that the CA Ideal programmer is aware of this key and the proper coding techniques.

Defining Dataviews with Datadictionary

This section summarizes the creation of dataviews using Datadictionary Online (DDOL). The Datadictionary Batch Utility DDUPDATE provides a mechanism for creating and maintaining dataviews and their related file model structures in batch.

A complete description of the information summarized here is provided in the appropriate Datadictionary documentation.

1. SET MODE DBMAINT or select option 1 on the mode menu.
2. Create the database in STAT Tnnn. You are prompted for the name of the areas and required attributes. When you enter and apply this information, DDOL displays the structure so far, consisting of just the database and areas.
3. Modify each AREA (for example, use MOD line commands and PF4). This prompts for the next level (table) attributes and gives an opportunity to modify those of the area. Entering the APPLY command shows the area structure so far, with a file and a table. If there are multiple areas, use PF4 to bring up the next area.
4. Modify the table and declare the keys and elements. DDOL provides only one line, but you can insert extra lines as needed. For each key, enter the attributes marked Key Only. Enter the APPLY command to see the file structure so far.
5. DEFINE the fields for the table, keys, and elements (using DEF line commands and PF4). DDOL processes the record definition. When the record definition is successfully applied, use PF4 to move on to the keys and type in the field name. Enter the APPLY command, and then press PF4 to move on to the element, where the Include is changed to “Y” for each field to include. Apply the elements.
6. Create the dataview (accessing elements). You do not need to define the keys here; CA Ideal finds the keys from the previously defined structure. Fill in the table name, the table's version, the elements, and whether the dataview is used for updating data. Enter the APPLY command.
7. If required, copy the database structure to status Prod. This verifies that the definition is consistent before the change is made.
8. Copy the dataviews to Prod if the database is now in PROD.
9. Catalog the database definition. This updates the CXX file.
10. Enable the database structure (TEST version or PROD).

11. Create and initialize the Index and Data areas.
12. In CA Ideal, use the CATALOG DATAVIEW command to catalog a dataview. A dataview must be cataloged before CA Ideal uses it. You can issue one or many CATALOG commands in a batch run of CA Ideal.

    When the CATALOG DATAVIEW command is processed, a DISPLAY DATAVIEW command is automatically invoked. If one or more rules are violated, a list of error messages displays at the bottom of the dataview display (or, if the CATALOG command is issued using CA Ideal in batch, a listing is printed).

    CATALOG DATAVIEW results in a cataloged dataview definition ready for CA Ideal use. Any CA Ideal program naming the dataview as a resource that compiles successfully can be executed.

    Access the CATALOG prompter by selecting option 4 on the CA Ideal Administration Maintenance Menu. For syntax and description of the command, see the Command *Reference Guide*.

Recatalog and Recompile

CA Ideal captures all information about a dataview, including field attributes, when the dataview is cataloged. This information is incorporated into the program when the program is compiled and actually becomes part of the program object module.

If you change the table structure that affects the dataview (for example, a change in the element length) or make a change that affects the attributes of any field in the dataview (such as class, type, length, decimal positions, occurs, value, or the relative position of the field in any element) or key structure (adding or deleting fields in key definitions), then you must:

1. Recatalog the database.
2. Enable the structure and, if necessary, the dataview.
3. Catalog the new dataview in CA Ideal.
4. Recompile all programs using that dataview.

You can perform the CA Ideal CATALOG DATAVIEW without performing the dictionary catalog database, but you cannot run programs against the dataview without performing the dictionary catalog.

You must enable the dictionary structure after performing any function that disables the structure (for example, COPY, ADD, or DEFINE), or an attribute maintenance function that modifies an occurrence.

If you change attributes of a PROD status dataview (for example, UPDATE), the dataview must be restored to TEST status to make the change. When the dataview is copied back to PROD status, a new VERSION is assigned. Follow the recatalog steps 1 to 4 described above, and UPDATE the resources of the programs that access the dataview.

If the table was changed, but not the dataview, the PROD dataview is connected to the new PROD versions of ELEMENTS. Follow steps 1 to 4 only if the updates affected the FIELD, ELEMENTs, or KEYs of the dataview.

When a dataview is copied in DDOL, both the old and new versions of the dataview are immediately reflected in the resource table of every program using the old version of the dataview. This is because the CA Ideal program resource table is a dynamic presentation of all the recognized relationships for a program at the time of the display (or edit).

When you COPY an entity-occurrence in DDOL, all of its relationships (except ordered relationships) are automatically copied too. Thus, since the dataview is related to every program for which it is a resource by the PGM-DVW-USE relationship, copying the dataview also copies each of these relationships. As a result, both the new PROD version and the previous version are related to every program. The resource table of a TEST status program shows both PROD and HIST status dataviews. You must manually remove the HIST status dataview. (This is not a problem with a program in PROD status.)

Chapter 4: Creating Dataviews for Sequential Files

CA Ideal supports dataviews for QSAM or SAM files created outside of CA Ideal. Sequential file dataviews in batch can be input or output. In CICS, they can be output only.

CA Ideal supports two types of sequential file dataviews-those that are modeled in Datadictionary and those that are not modeled.

- Modeled dataviews are created and maintained in Datadictionary. Their fields, records, and relationships with programs and systems are defined in the dictionary. They are available only to CA Datacom/DB sites.
- Unmodeled dataviews are created and maintained in CA Ideal. Their definitions are stored in the Virtual Library System (VLS). They are available to all sites.

This chapter describes how to create and maintain dataview definitions for both types of sequential files.

Defining Modeled Sequential Files

The Data Administrator uses Datadictionary in batch or online mode to maintain file and dataview definitions for modeled sequential files. For modeled dataviews, there are no facilities to describe and maintain descriptions in CA Ideal.

Data Structures

This section describes the data administration functions of the CA Ideal user. It assumes some familiarity with the Datadictionary. It explains how to establish and maintain modeled dataview definitions for sequential files that CA Ideal uses. See the CA Datacom/DB documentation for a full explanation of terms and use.

Since dataview definitions are stored in Datadictionary, your DA should establish and maintain them using Datadictionary. Step by step instructions on defining a Datadictionary are provided in the section titled Defining Modeled Dataviews for Sequential Files in this chapter. However, to be sure you are creating a valid dataview for CA Ideal; you should familiarize yourself with the rules that CA Ideal enforces when cataloging a dataview before defining the dataview in the dictionary.

The functions of a Data Administrator include:

- Defining the file layout in the dictionary, including establishing fields.
- Defining elements of the record in the dictionary. An element is a group of one or more contiguous fields in a record.
- Defining dataview definition occurrences in the dictionary facility.

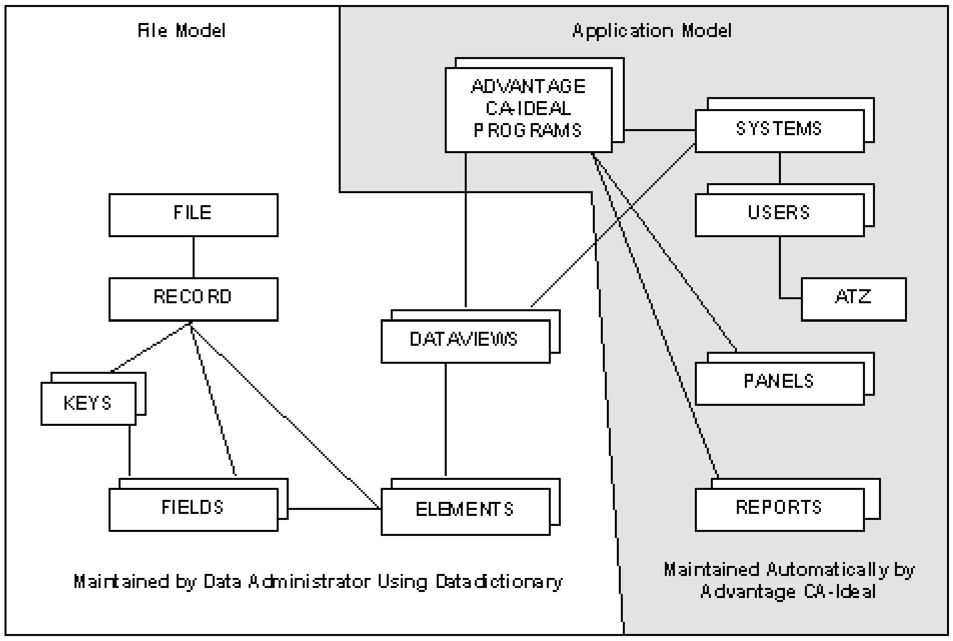
Access Datadictionary Online by selecting option 3 on the Administration Maintenance Menu or by issuing the CA Ideal command:

```
DDOL
```

At any time during the use of DDOL, the DA can return to the CA Ideal Main Menu by issuing the DDOL command:

```
IDEAL
```

The following graphic illustrates the entity types and relationships in the dictionary and their relationship to CA Ideal. In this graphic, the unshaded area represents the file model, and the shaded area represents the application model.


File Model
Application Model
ADVANTAGE CA-IDEAL PROGRAMS
SYSTEMS
USERS
ATZ
FILE
RECORD
DATAVIEWS
PANELS
KEYS
FIELDS
ELEMENTS
REPORTS
Maintained by Data Administrator Using Datadictionary
Maintained Automatically by Advantage CA-Ideal

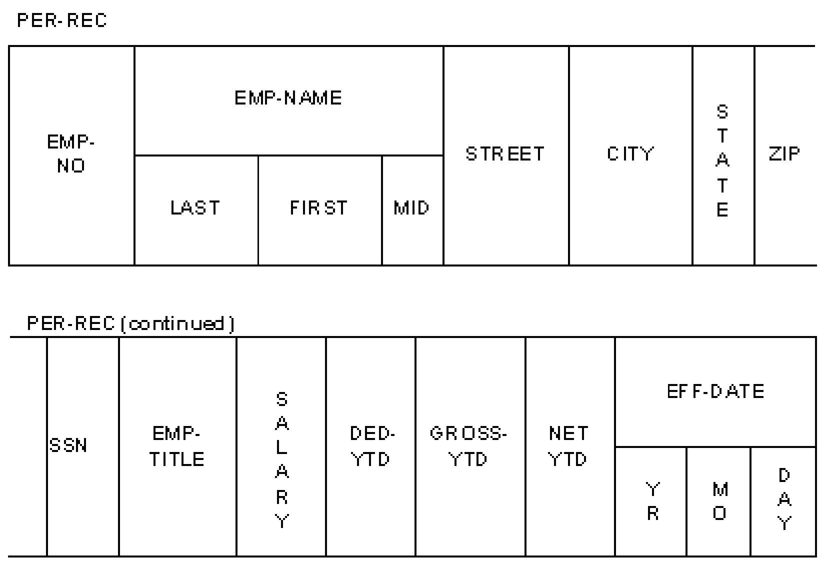
Notice that the file model for a sequential file is different from the model for an CA Datacom/DB table. The AREA and TABLE are replaced by a FILE and a RECORD and no DATABASE is specified.

File Model

The process of defining the file and record layout is fully explained in the CA Datacom/DB documentation in the section on FILEMAINT MODE. In addition to file model information required by the dictionary facility, the Data Administrator should refer to the CA Datacom/DB documentation for descriptions of utilities for maintaining the dictionary definitions.

The file model for sequential dataviews must include all entities in the dictionary file model starting at the FILE entity and going down the structure.

The following graphic is a record layout as defined in the dictionary facility for a modeled sequential file.



Defining Elements

In the dictionary, a sequential file dataview is defined as a single element, with an element comprising all fields in a record. Therefore, the element defined for the PER-REC file contains all the fields shown in the previous figure. CA Ideal imposes restrictions that limit both the elements that can be related to a given dataview for CA Ideal and how you can use those elements.

The following restrictions apply for dataviews and their related elements that CA Ideal accesses:

- An element related to a CA Ideal dataview must be explicitly defined as a collection of contiguous fields. In particular, when performing the DEFINE ELEMENT or the MODIFY ELEMENT function in the DD statement, once the starting and ending fields are selected, all intervening fields must be explicitly included in the element. If fields are added to the file (when it is in TEST status), the additions are not automatically reflected in the file's related elements. The Data Administrator must also execute DEFINE ELEMENT or MODIFY ELEMENT and explicitly include the new fields for CA Ideal if they are in the range of an existing element that CA Ideal uses.
- If a compound (group) field is included in an element, all of its subordinate fields must also be explicitly included in the element. If a field is redefined, not all redefinitions need to be explicitly included. However, the redefined field must be explicitly included if any of the redefining fields are included.

Rules for Sequential File Dataviews

CA Ideal imposes certain rules on dataviews used for sequential files. The CA Ideal CATALOG command enforces the rules. For more information, see the *Command Reference Guide*.

Dataview Status and Related Occurrences

A dataview can be successfully cataloged in TEST or PROD status. A dataview that is cataloged in TEST status has all elements in TEST status and all version numbers of the elements are the same as the dataview version. A dataview that is cataloged in PROD status has all elements in PROD status. The elements in PROD status do not have to have the same version number as the dataview. The dataview must be related to one element. The element must be related to a single record, and the record to a file.

Note: Exceptions exist for dataviews created before Datadictionary r2.4. Refer to Datadictionary documentation.

Dataview Attributes

For a dataview that CA Ideal accesses, the only DATAVIEW attribute whose specified value is used is UPDATE-INTENT. You can specify Y for updateable or N for non-updateable. (U for updateable is also supported for compatibility with earlier releases.) CA Ideal interprets any other value as N, non-updateable. If the DATABASE entity-occurrence name is DATA-DICT or UNIVERSAL, CA Ideal assumes a value of N, non-updateable.

The attributes DBMS-USED and ACCESS-METHOD for the dataview attribute are updated with values CA Ideal supplies when the dataview is cataloged.

FILE Attributes

For CA Ideal to access a sequential dataview, FILE attributes must be defined as follows:

- DBMS-USED must be NONE
- ACCESS-MTHD must be QSAM or SAM
- RECORD-FORMAT must be FBLK or FUNB
- MONITOR-NAME must be supplied (see following)
- Under VSE, the LABEL, DEVICE, MAX-RECORD-SIZE, and MAX-BLOCK-SIZE attributes must be specified for SAM files

This information is summarized in the following table.

	z/OS	**VSE**
DBMS-USED	NONE	NONE
MAX-RECORD-SIZE	Actual rec size	Actual rec size
MAX-BLOCK-SIZE	See the section Runtime Considerations for Sequential File Dataviews in this chapter	See the section Runtime Considerations for Sequential File Dataviews in this chapter
DEVICE	See the section Runtime Considerations for Sequential File Dataviews in this chapter	TAPE|PRT|READER, PUNCH|2314|3330, 3330II|3340|3350, |3375
ACCESS-MTHD	QSAM	SAM

	z/OS	VSE
MONITOR-NAME	Batch: DDNAME Online: CICS DESTID	Batch: DISK - DLBL Name SLTAPE - TLBL Name Other - any identifier Online: CICS DESTID
LABEL	See the section Runtime Considerations for Sequential File Dataviews in this chapter	Y|N (DEVICE=TAPE only)
RECORD-FORMAT	FUNB|FBLK	FUNB|FBLK

For more information, see Run-Time Considerations for Sequential File Dataviews in this chapter.

Sequential File MONITORNAME

For a sequential file dataview, the MONITOR-NAME attribute of the FILE entity-occurrence must contain a logical data set identifier. (The term MONITOR-NAME indicates that this is the logical name by which the file is known to the host monitor.) In batch, the host monitor is actually the host operating system. In a given run, each dataview can have a unique MONITOR-NAME relating each dataview to a separate physical data set. Or two (or more) dataviews can have the same MONITOR-NAME value, which causes the two dataviews to share the same physical dataset (for output files, these results in interleaved output).

Under z/OS or VSE

All runs in a CICS partition that use the same sequential dataview for output write interleaved records to the same physical file. In addition, MONITOR-NAME connects the dataview to the host environment in the following ways:

- **CICS** The first four characters of the MONITOR-NAME must appear as the DESTID in a DFHDCT entry. This DESTID relates the dataview to the JCL as with any CICS DFHDCT.
- **Batch (z/OS)** The MONITOR-NAME becomes the ddname for the file and must conform to the rules for z/OS ddnames.
- **Batch (VSE)** The MONITOR-NAME becomes the DLBL name for DISK files or the TLBL name for standard label tape (SLTAPE) files and, in these cases, must conform to the rules for VSE DLBL and TLBL names. For all other device types, MONITOR-NAME is not used in the JCL.

Variable Length Records

Not supported.

Element Consisting of Contiguous Fields

For a sequential dataview, there can be only one element. It must explicitly include all the fields in the record. It is possible to accidentally violate this rule by inserting a field through DEFINE FILE, but forgetting to include the field in the appropriate DEFINE or MODIFY element.

Levels of Data Structure Nesting

The maximum number of nested levels of groups and fields in a dataview is 15. Since the dataview itself is the outermost level (level 1), the highest level number allowed is 16. Since the dictionary assigns level 1 to all columns with a PARENT of START, the level numbers that CA Ideal assigns is always at least one greater than the level numbers assigned by the dictionary.

Number of Levels of Occurrences

CA Ideal supports a maximum of three nested levels of occurring fields (that is, tables of up to three dimensions are supported).

Compound Fields

Compound fields (group fields) should not have initial values. If they do, the value is ignored and a warning message issued.

Data Types of Fields

CA Ideal supports the following data types:

- Alphanumeric (display) (C)
- Zoned decimal (N)
- Packed decimal (D), and
- Binary (B)

CA Ideal converts all other unknown data types to alphanumeric and issues a warning message.

CA Ideal supports numeric fields up to the equivalent of 31 decimal (base 10) digits (for zoned or packed decimal) and up to the equivalent of 9 decimal (base 10) digits (full word or half word binary) for binary fields. CA Ideal supports only half word (2 bytes) and full word (4 bytes) binary fields. You can define binary fields of other than 2 or 4 bytes (including data types 2, 4, and 8), but CA Ideal treats them as alphanumeric fields (with a warning message) and the user must call a non-CA Ideal subprogram to handle them. CA Ideal does not support floating point items of any precision.

Other Field Attributes

CA Ideal ignores FIELD occurrences with CLASS=I (INDEXED BY). The BLANK-WHEN-ZERO attribute is also ignored. If either the LOW-RANGE or HIGH-RANGE attribute is used with a CLASS=V FIELD, a warning message occurs.

The REDEFINES attribute has the following values:

- **N** (Does not redefine) is the default
- **P or S** Treated as P (redefines the previous FIELD occurrence at the same level)

Edit Patterns

You can specify any valid COBOL edit pattern (except Sterling currency symbols) for a field in the dictionary (attribute EDIT-PATTERN). It is used in CA Ideal reports unless overridden by RDF (see the *Generating Reports Guide*). PDF also uses the edit pattern specified in the dictionary for dynamically generated panel fields.

Field Column Headings

You can specify field column headings for fields in the dictionary (attributes HEADING-1 and HEADING-2). However, headings longer than 20 characters are truncated to 20 characters in the CA Ideal cataloged dataview. PDF uses the field column headings specified in the dictionary for dynamically generated panel fields. They are also used in CA Ideal reports unless overridden by RDF.

Note: CA Ideal uses a not-sign (⍰) as the begin-line character in report headings. For more information, see the *Generating Reports Guide*.

PDL Reserved Words

You cannot use a PDL reserved word as the name of a dataview (see the reserved word list in the *Command* Reference *Guide*). You can use reserved words for field names; however, qualification of such field names with the dataview name is required. For example, assume there is a dataview called SHIPMENT with elements containing fields called TO and FROM. It is valid for a PDL procedure to have statements:

```
MOVE SHIPMENT.TO TO LOCATION
SUBTRACT SHIPMENT.FROM FROM AMOUNT
```

However, the following command is ambiguous and causes a program compilation error.

```
MOVE TO TO LOCATION
```

Use of Field Entity Names

For names of fields, CA Ideal uses the 32-character field entity-occurrence name, not the 30-character compiler name. Field names in CA Ideal must be unique in the dataview. This occurs automatically since all elements must belong to one record and the dictionary requires field names to be unique in the record name.

MultiValue Condition Names

CA Ideal supports one unique value for each condition name (class V field in the dictionary). Lists or ranges of values are not supported.

Figurative Constants

CA Ideal supports the following figurative constants as values in the dictionary:

SPACE
SPACES
ZERO
ZEROS
ZEROES

The VALUE attribute for a condition-name (class V field in the dictionary) cannot be blank. A value of all spaces can only be represented by the figurative constants SPACE or SPACES.

Optional FIELD Attributes

CA Ideal recognizes the following considerations for optional FIELD attributes:

- **SEMANTIC-TYPE** Entered for date fields only. Enter a SEMANTIC-TYPE of CA-DATE for a numeric field (TYPE=N, D, or B) that contains an internal date stored as the number of days since (positive number) or preceding (negative number) the date January 1, 1901. A five-digit number can contain values from -99,999 through 99,999 or the equivalent of almost 274 years before and after the base date. This is from approximately the year 1627 to approximately the year 2175. A six-digit number can represent dates approximately 2740 years before and after the base date, and so on. SQL semantic types may not be used for sequential dataviews.
- **DECIMALS** Total number of decimal (fractional) places in the number. CA Ideal determines the number of integer places by calculating the total number of decimal (base 10) digits that fit in the field (based on the TYPE) and subtracting the DECIMALS value.
- **DEPENDING ON** Name of a non occurring simple numeric field with DECIMALS=0 that precedes this field in the record. The Advantage
CA-Ideal restriction that a field with a DEP ON clause can be a simple or compound field, but must not be followed in the dataview by any other fields (unless they are subordinate fields to a compound field).
- **EDIT-PATTERN** COBOL-like edit pattern used when printing the field. The CA Ideal Report Definition Facility and the CA Ideal Panel Definition Facility use this pattern as the default for the field. If you omit this edit pattern, the following default edit pattern is used:
  - **X(n)** Alphanumeric fields (where *n* is the length)
  - **9(i).(9d)** Numeric fields (where *i* = integers, *d* = decimals)

- **HEADING-1 and HEADING-2** One or two literals (without quotes) used as the default column headings for the CA Ideal Report Definition Facility and for dynamic field definition using the CA Ideal Panel Definition Facility. For example, if ACCOUNT is specified as the value for HEADING-1 and NUMBER is specified as the value for HEADING-2, the column header appears as follows:

  ```
  ACCOUNT
  NUMBER
  ```

- **SIGN** N to force “unsigned” (the default for numeric fields is Y).

- **VALUE** For numeric fields, a valid numeric literal or one of: ZERO, ZEROS, ZEROES.

  For alphanumeric fields, enter the characters to appear as the initial value without quotes (or any other delimiter) or enter one of: SPACE, SPACES.

  For a field with CLASS=V (value, or condition-name in CA Ideal), you must type at least one non-blank character for VALUE. If the condition-name corresponds to a value of spaces, type the word SPACE or SPACES.

  If a value contains embedded blanks, type in the value with no delimiters. For example, enter a three-character value “A B” as A in position 1, blank in position 2, B in position 3.

  CA Ideal does not support any of the following:

  - Multiple values for a single condition-name.
  - A range of values for a field, either in the VALUE attribute or the LOW-RANGE and HIGH-RANGE attributes.
  - COBOL figurative constants:
    - HIGH-VALUE(S)
    - LOW-VALUE(S)
    - QUOTE
    - ALL literal
  - NULL-INDICATOR is not supported for sequential dataviews.

Modeled Sequential File Structures and Dataviews

In CA Ideal, a dataview of a sequential file appears as a collection of fields comprising a record in a sequential file. Internally, the dataview is an entity-occurrence related through the DVW-ELM-ACCESS relationship to a single element comprising the record for the file.

The DBA must create the file model entity-occurrences and the dataview entity-occurrence and relate them for successful cataloging, compilation, and running in CA Ideal. The Datadictionary documentation describes the steps the Data Administrator follows in defining the database structure and dataviews.

This section summarizes the creation of dataviews using DDOL (Datadictionary Online) FILEMAINT mode. You can also use the Datadictionary Batch Utility, DDUPDATE, to create and maintain dataviews and their related file structure in batch. The Datadictionary documentation provides a complete description of the information summarized here.

For steps required before execution, see Runtime Considerations for Sequential File Dataviews in this chapter.

Defining Modeled Dataviews for Sequential Files

1. Enter SET MODE FILEMAINT or select option 5 on the mode menu. Do not attempt to create an unmodeled sequential file dataview in any other mode.
2. Create the file. You are prompted for the record name and required attributes. When you enter the APPLY command, the file structure so far displays.

   Note: The CA Ideal requirements for FILE attributes are described in the section titled Rules for Sequential File Dataviews.
3. Modify the record and declare the KEY and ELEMENT. Insert an extra line because DDOL gives you only one. Supply all attributes marked Key Only for the key. Enter the APPLY command to see the file structure so far.

   Note: CA Ideal does not use the KEY information. CATALOG DATAVIEW processing ignores it. You can define a key to document the native sequence of the file without affecting CA Ideal.
4. Define the fields for the RECORD, KEYs, and ELEMENTs (using DEF line commands and PF4). DDOL first processes the record definition. When that is successfully applied, type PF4 to move on to the key, where the field name must be typed in. Enter the APPLY command, then type PF4 to move on to the element. Change the Include attribute to Y for each field. If there are no redefinitions, you can alternatively name the first and last fields. Enter the APPLY command for the element.
5. Create the dataview (accessing elements). CA Ideal does not need a key defined here. The dataview must be created in the same status and version as the file. Fill in the name of the file and element and enter the APPLY command.
6. If required, copy the file structure to status PROD. This verifies that the definition is consistent before it actually makes the change.
7. Copy the dataview to PROD if the file is now PROD.
8. Enable the file structure (test version or PROD).

In CA Ideal

Use the CATALOG DATAVIEW command or equivalent CATALOG prompter to catalog a dataview. A dataview must be cataloged before CA Ideal uses it. You can issue one or many CATALOG commands in a batch run of CA Ideal.

When the CATALOG DATAVIEW command is processed, a DISPLAY DATAVIEW command is automatically invoked. If one or more rules are violated, a list of error messages displays at the bottom of the dataview display (or, if the CATALOG command is issued using CA Ideal in batch, a listing is printed).

CATALOG DATAVIEW results in a cataloged dataview definition ready for CA Ideal use. Any CA Ideal program naming the dataview as a resource that compiles successfully can be executed.

Access the CATALOG prompter by selecting option 4 on the CA Ideal Administration Maintenance Menu. For more information about description and syntax of the command, see the *Command Reference Guide*.

Recatalog and Recompile

CA Ideal captures all information about a dataview, including field attributes, when the dataview is cataloged. This information is incorporated into the program when the program is compiled and actually becomes part of the program object module.

If you change the file model that affects the dataview (for example, a change in the element length) or make a change that affects the attributes of any field in the dataview (such as class, type, length, decimal positions, occurs, value, or the relative position of the field in any element), then you must:

1. Enable the file structure and dataview.
2. Catalog the new dataview in CA Ideal.
3. Recompile all programs using that dataview.

You must enable the structure after performing any function that disables the structure (for example, COPY, ADD, DEFINE) or an attribute maintenance function that modifies an occurrence.

If you change attributes of a PROD status dataview (for example, UPDATE), you must restore the dataview to TEST to make the change. When the dataview is copied back to PROD status, a new version number is assigned. Follow the recatalog steps 1 to 3 above and update the resources of the programs that access the dataview.

When a dataview is copied in DDOL, both the old and new versions of the dataview are immediately reflected in the RESOURCE component of every program using the old version of the dataview. This is because the CA Ideal program resource panel is a dynamic presentation of all the recognized relationships for a program at the time of the display (or edit).

When you copy an entity-occurrence in DDOL, all of its relationships (except ordered relationships) are automatically copied also. Thus, since the dataview is related to every program for which it is a resource by the PGM-DVW-USE relationship, copying the dataview also copies each of these relationships. As a result, both the new PROD version and the previous version are related to every program. The resource table of a TEST status program shows both PROD and HIST status dataviews. You must manually remove the HIST status dataview. (This is not a problem with a program in PROD status.)

Defining Unmodeled Sequential File Dataviews

CA Ideal provides a facility for creating, describing, and maintaining dataview definitions for sequential files without a file model. An unmodeled sequential dataview is recorded as a DATAVIEW entity occurrence in the dictionary. The definition of the fields comprising the dataview is recorded in a VLS member.

The dataview is, like other CA Ideal dataview types, a logical view of the specified fields that can be shared across many applications. The CA Ideal user sees the data as a named collection of fields in a table. PDL statements for accessing data through dataviews apply. You can find any record or collection of records in the table by using the PDL FOR statement in a PDL procedure definition.

The dataview definition is created and cataloged in CA Ideal. It can be displayed, edited, duplicated to a new name or a new version, printed, and deleted.

The unmodeled sequential file dataview is unlike other dataview types in several ways:

- It is created in CA Ideal and can be edited, duplicated to a new version or a new name, and deleted in CA Ideal. The test version of the dataview can be edited, even when it is used as the resource of a production program. You cannot edit the production version of the dataview. However, you can duplicate it to a new version, edit it, and recatalog it. You also need to recompile all programs that use the dataview.
- You can limit access to the dataview to a particular CA Ideal system only if you are an CA Datacom/DB site.

Components of a Sequential File Dataview Definition

A CA Ideal sequential file dataview definition uses the following components:

- An *identification* fill-in that initiates the creation of a sequential file dataview definition and provide descriptive information about it.
- A *field definition* fill-in that describes the fields comprising the dataview.
- A *parameter definition* fill-in that describes the file associated with the dataview.

This section contains detailed explanations of how to complete the fill-ins that construct each component of a sequential file dataview definition presented in the Dataview Menu.

The functions CA Ideal provides to define and maintain sequential file dataview definitions are presented in the Dataview Menu. To access this menu, select option 2 on the Main Menu or enter the DATAVIEW command. See Dataview Menu in the “Introduction” chapter for instructions on using this menu.

Creating a Dataview Definition

The first step in creating an unmodeled sequential file dataview is identifying the dataview. To identify the dataview, you must first enter the CREATE DATAVIEW command, and then fill in the Dataview Identification screen. Once you name the dataview either in the CREATE DATAVIEW command or on the Identification fill-in, an entry for the dataview definition is stored in the dictionary facility. After the dataview is identified, it becomes the current definition.

The CREATE DATAVIEW command presents a blank dataview identification fill-in. This fill-in establishes the dataview name and provides identification information for the dataview that is kept in the dictionary. CREATE DATAVIEW creates version one in test status. You can modify, duplicate, print, delete, and mark this version to production status, just like any other CA Ideal entity.

Identifying the Dataview

Use the CREATE DATAVIEW command to display the dataview identification fill-in for a new dataview. To display the Identification fill-in for an existing dataview:

- If the dataview is not the current definition, enter the EDIT or DISPLAY DATAVIEW command, specifying the IDENTIFICATION component in the command.
- If the dataview is the current definition, enter the IDE command or press the equivalent PF key.

The dataview identification fill-in shown below enters descriptive information about the sequential file dataview, either when the dataview definition is initially created or when it is subsequently modified.

```
=>
=>
=>
-------------------------------------------------------------------------------
 IDEAL: DVW IDENTIFICATION  DVW              SYS: DOC  FILL-IN

DATAVIEW Name  __________________
Access-Method  ____ (SEQ|VSAM)
Operating System MVS (MVS|VSE)



Short Description ________________________
Description:
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
```

The components of the dataview identification fill-in are as follows:

Dataview Name

Specifies the name, one- to 18-character, assigned to the sequential file dataview definition. CA Ideal initializes the name to the name entered in the CREATE command or prompter, if a name was supplied.

Access Method

Type of file being defined: VSAM for a VSAM file, SEQ for a sequential file. Once the access method is entered and accepted, you cannot modify this value.

Operating System

Specifies the type of operating system under which the dataview is accessed. The default value for this field is the current system.

Created By

Initial creation date of the dataview definition and the user ID of the creator. This information appears only after the name is entered and accepted.

Last Modified ...at ...

Identifies the Date, time, and user ID of the last edit access. This is blank until an EDIT DATAVIEW command accesses this definition. This information appears only after the dataview name is entered and accepted.

Last Cataloged ...at ...

Identifies the Date, time, and user ID of the last catalog.

Short Description (Optional)

Specifies the brief description of the dataview definition the user entered.

Description (Optional)

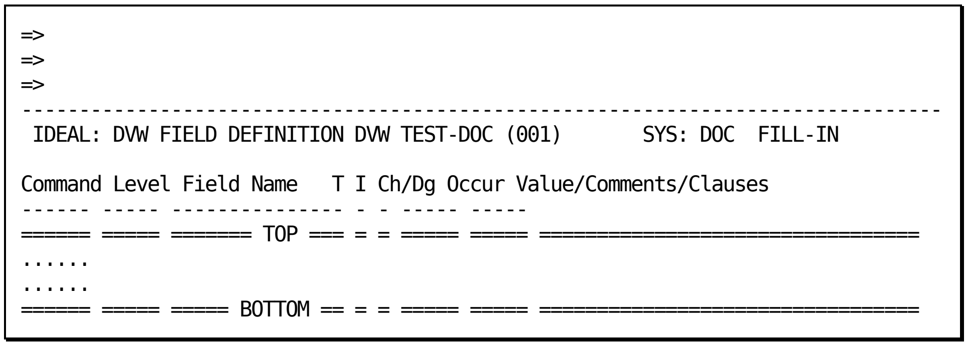
Specifies the full description of the dataview definition, up to five lines long.

The dataview identification fill-in is shown as follows:

```
=>
=>
=>

-------------------------------------------------------------------------------
 IDEAL: DVW IDENTIFICATION  DVW EMPLOG  (001) TEST     SYS: DOC  FILL-IN

DATAVIEW Name  EMPLOG
Access-Method  SEQ (SEQ|VSAM)
Operating System MVS (MVS|VSE)

Created     10/26/93        By KRITZ
Last Modified  10/25/94 at 08:38   By DESANDRES
Last Cataloged  10/26/94 at 10:45





Short Description Employee Log File______
Description:
Log file to track transactions against the Employee Master File
_______________________________________________________________
_______________________________________________________________

_______________________________________________________________
_______________________________________________________________
```

Defining Fields

The FIELD command or equivalent PF key displays the dataview field definition fill-in for the current sequential file dataview definition. The dataview field definition fill-in, shown in the following screen, names and describes the fields in the dataview.



```
=>
=>
=>
-------------------------------------------------------------------------------
 IDEAL: DVW FIELD DEFINITION DVW TEST-DOC (001)      SYS: DOC  FILL-IN

Command Level Field Name   T I Ch/Dg Occur Value/Comments/Clauses
------ ----- -------------- - - ----- -----
====== ===== ======= TOP === = = ===== ===== ================================
......
......
====== ===== ===== BOTTOM == = = ===== ===== ================================
```

The entries on the dataview field definition fill-in are as follows:

Command

Specifies the area where you can specify line commands. For more information about description of line commands, see the *Command Reference Guide*.

Level

Specifies the number that hierarchically ranks fields. Levels start with level 2. (The dataview is treated as a level-1 item.) Level-2 elementary fields (fields with no subordinate fields) do not require a level number. If no level number is supplied, the level number defaults to 2.

Field Name

Field Name column must contain one of the following:

- A valid name that identifies an elementary or group field. See the *Programming Reference Guide*.

  An elementary field is a field that has no subordinate fields. The field name of an elementary field must be unique in the dataview. Elementary fields are further defined by values entered in the T and Ch/Dg field. A group field has subordinate fields. Group field name entries have no T and Ch/Dg values.

  You can continue the field name on a second line by including a semicolon (;) as the last character on the first line. You can break the field name at any character. You can specify the level number and the attributes T, I, Ch/Dg, and Occur on the first line only.

- A condition name for a condition name definition.
- A valid subscript of the form (n), (n,o), or (n,o,p) for an initial value of a specific occurring item.
- A filler, an unnamed field that reserves space. If the field is unnamed, the field name column contains blanks.
- Blanks, when the Value/Comment/Clauses column contains a continuation line, a comment, or when the rest of the line is blank.

T (Type)

The field types are as follows:

- **X** Alphanumeric field. The value of the field can contain alphabetic, numeric, and special characters.
- **N** Signed numeric field. The value of the field can contain 0-9, a minus sign, and a decimal symbol.
- **U** Unsigned numeric field. The value of the field can contain 0-9 and a decimal symbol.
- **D** Date field. This type is a numeric field containing an integer number, reflecting the number of days, plus or minus, from December 31, 1900 (day zero). This is not the same as the Date data type in SQL dataviews.
- **C** Condition name assigned to a specific value of a field.

Note: Type defaults to N if you specify the internal representation. Type must be blank for a group item. A detailed explanation of each field type and its characteristics is provided in the *Programming Reference Guide*.

Variable length fields are not supported for sequential files.

Non-date entry with no type or length information is assumed to be a group name and must have subordinate fields following it.

I (Internal Representation)

Internal representation of numeric (signed and unsigned) and date type fields:

- **P** Packed
- **Z** Zoned
- **B** Binary

Note: Internal representation must be blank unless the type is N, U, or D. P is the default.

CH/DG (Characters/ Digits)

The length of the field value:

- **CH** Number of alphanumeric characters in the field value.
- **DG** Sum of the number of integer and decimal places in a numeric or date field value, separated by a decimal point. (Date fields cannot have decimal places.)

  You must specify the characters and digits for all elementary field types except dates and flags. The default for date fields is 7. The minimum length for date fields is 5. For numeric and date fields, the maximum is 31 for packed and zoned and 9 for binary.

  A non-date entry with no type or length information is assumed to be a group name and must have subordinate fields following it.

  In the following illustration, 42 in the Characters/Digits column for the first alphanumeric field (Type X) indicate a length of 42 characters. A numeric field (Type N) with 7 specified in the Characters/Digits column indicates a seven-digit field with seven integer positions. The next numeric field, with 10.3 specified in the Characters/Digits column, indicates a 13-digit field with ten integer positions and three decimal places. A date field (Type D) with five characters/digits can hold an internal 5-integer date value of up to 273 years from the base year.

Type	CH/DG
X	42
N	7
N	10.3
D	5

Occur (Number of Occurrences)

Specifies the number of times a group or field occurs. You can specify initial values for the individual occurrences of an occurring field by entering the values on each successive line following the occurring field definition. A valid subscript number (of the form (n), (n,o), or (n,o,p)) is entered on the corresponding line in the Field Name column.

You cannot specify initial values for repeating groups.

Value/Comments/Clauses (Value, Comments, or Redef)

Depending on the circumstances, this column specifies a value for the occurrence of the field or a REDEFINITION keyword. You can specify a descriptive comment about the field alone or with any of the others.

Note: The case of the data entered in this column as CA Ideal retains it depends on the setting that was determined for case either by default or with a SET EDIT CASE command.

Value

Specific value assigned to an occurrence of a field.

You can assign a value to an elementary field in the Value/Comments/Clauses column. This value can be a numeric or alphanumeric literal, depending on the type of the elementary field.

You must enclose alphanumeric literals in delimiters (“or “). If an alphanumeric literal is longer than the space provided, continue the alphanumeric literal on the next line of the same column surrounded by a new pair of delimiters. Leave all other columns on the continuation line blank. Two or more delimited alphanumeric literals continued in this fashion are concatenated and treated as one. Be sure to include any required spaces in the delimiters, since spaces are not added when the values are concatenated.

The default value for a numeric field is zero. The default for an alphanumeric field is spaces.

In the case of an *occurring field*, the Value column is left blank on the line that contains the field name for the parent field. The following lines in the Value column can contain entries for the value of each occurrence of the field.

You cannot assign an initial value to a *date field*. (The default initial value is 0.) This also means that you cannot define a date field with subordinate condition names or flags. However, a date field can redefine a numeric field with an initial value.

Comments

Useful information about the field. A comment is indicated in this column by a preceding colon (:). To continue a comment over multiple lines, start each line of the column with a colon. The other columns can be blank or the continuation of a field name.

Redefinition

Keyword REDEFINITION (or REDEF) in this column indicates that this item is another view of the closest previously defined item at the same level that is not itself a redefinition.

This item cannot be larger than the item it redefines. The two items can be different types (such as alphanumeric and numeric).

If a field with a REDEF is a subordinate field, its Type does not affect whether the group is alpha or non-alpha. CA Ideal uses the Type of the redefined (original) item. For more information about alpha and non-alpha group fields, see the *Command Reference Guide*.

A field with a REDEF cannot have an initial value. If a group has a REDEF, none of its subordinate fields can have initial values. (However, the original field can have an initial value.)

The following example illustrates the various types of data structures that you can define using the dataview field fill-in.

```
=>
=>
=>
--------------------------------------------------------------------------------
 IDEAL: DVW FIELD DEFINITION DVW TRANS-FILE (001)      SYS: DOC  FILL-IN

Command Level Field Name     T I Ch/Dg Occur Value/Comments/Clauses
------ ----- --------------- - - ----- ----- --------------------------------
====== ===== ======= TOP === = = ===== ===== ================================
000006   2   REC-TYPE        X       2
000007   2   DATA            X      98
000008   2   CUST-ADDR                         REDEF
000009   3   CA-CUST-ID      X       5
000010   3   CA-NAME         X      30
000011   3   CA-ADDRESS      X      30
000012   3   CA-STATE        X       2
000013   3   CA-ZIP          N Z     9
000014   3   CA-CITY         X      15
000015   2    ORD-INFO                          REDEF
000016   3   OI-ORD-ID       X       5
000017   3   OI-CUST-ID      X       5
000018   3   OI-ORD-DT       X      10
000019   3   OI-DISC-PCT     N P   2.1
000020   3   OI-STAT         X       1
000021   3   OI-SHIP-DT      X      10
000022   3   OI-TERMS        X      15
000023   3   OI-CUST-PO      X      10
000024   3   OI-SHIP-ID      X       5
000025   3   OI-ORDER-TOTAL  N P   7.2
000026   3   OI-FRT-AMT      N P   7.2
====== ===== ===== BOTTOM == = = ===== ===== ================================
```

Defining the File Parameters

The dataview parameter definition describes the sequential file corresponding to the dataview as follows. Use the PARAMETER command or equivalent PF key to display this fill-in for the current sequential file dataview definition.

```
=>
=>
=>

------------------------------------------------------------------------------
 IDEAL: DVW PARAMETERS    DVW TEST-DOC (001)       SYS: DOC  FILL-IN

Update Intent    Y     (Y=Yes, N=No)
Filename      SEQDVW1
```

The components of the dataview parameter fill-in are as follows:

Update Intent

Y Dataview can update the table.

N Dataview cannot update the table.

Filename File name from the CA Ideal startup JCL:

z/OS One- to eight-character ddname from the DD statement.

VSE One- to seven-character DLBL or TLBL name from the DLBL or TLBL statement.

For VSE Only

The following components display when the operating system is VSE:

Record Length Required. It must be from 1 to 32,000.

Block Size Required. It must be a multiple of the record length and cannot exceed 32,000.

Record Format One of the following:

FBLK Fixed block

FUNB Unblocked

Device Type One of the following:

DISK TAPE

PRT

PUNCH

Tape Labels One of the following:

Y Tape labels

N No tape labels

Runtime Considerations for Sequential File Dataviews

When executing a CA Ideal program with a sequential file dataview, the considerations covered in this section apply.

Descriptions for: Online versus batch processing, CICS, z/OS batch, VSE batch, and modifying dataview characteristics are covered in this section.

Online Versus Batch Processing

In batch, the job controls the sequential files. They are opened at the FOR statement and closed at the ENDFOR statement. However, in the online environment, CA Ideal cannot control sequential files because multiple users might be accessing the files at the same time. To ensure file integrity, CICS controls sequential files accessed by transactions running online. This means that CICS opens and closes sequential files that CA Ideal uses online and, until CICS closes the files, they cannot be read in batch.

CICS

Setting up the environment for writing to a sequential file from one or more CICS CA Ideal applications is described in this section. When more than one CICS user writes to the same sequential file, the records are interleaved. This makes it possible, for example, to log records from various users to a common log file.

Note: In CICS, CA Ideal can write to sequential files but cannot read sequential files. Until CICS closes a sequential file, it cannot be read in batch.

The steps for defining the sequential file are as follows:

1. Define the sequential file and a corresponding dataview and define it as a resource for the CA Ideal programs that uses it (see Defining Modeled Sequential Files and Defining Unmodeled Sequential File Dataviews earlier in this chapter).

   For dataviews modeled in the dictionary, make sure that the ACCESS-METHOD is defined as QSAM in the FILE entity-type, and that the MONITOR-NAME has a value in the first four characters, which corresponds to a CICS DCT DESTID. Make sure that the DATAVIEW has an UPDATE-INTENT of Y (yes).

   For unmodeled dataviews, make sure that in the dataview parameter fill-in the access method is defined as QSAM and that the filename has a value in the first four characters that corresponds to a CICS DCT DESTID. For example, a file called PAYFILE is defined with a MONITOR-NAME of PAYR and a corresponding DATAVIEW is defined with a name of PAYLOG. The dataview name PAYLOG is used in the CA Ideal program to add records to it as follows:

   ```
   FOR NEW PAYLOG
   MOVE ... TO PAYLOG.field
   ENDFOR
   ```

2. Define the file to CICS as a Transient Data Queue using RDO.

   For a dataview modeled in the dictionary, the queue name must match the MONITOR-NAME attribute, and for unmodeled dataviews, it must match the first four characters of the filename.

You must add a DD statement (z/OS) or DLBL or TLBL statements (VSE) for the file to the CICS startup deck.

z/OS Batch

Add a DD statement to the CA Ideal batch JCL for the file using the specified MONITOR-NAME or filename.

The z/OS IDSYSFT contains 16 entries of the form SYSUT01 through SYSUT16. CA Ideal uses these entries for sequential files, actually modifying the file name dynamically. For this reason, no changes to the z/OS IDSYSFT (batch file table) are required.

The maximum number of active sequential files at any point in time is 15.

VSE Batch

For a standard label tape file, add a TLBL statement to the batch CA Ideal JCL using the specified MONITOR-NAME or filename. For a disk file, add a DLBL statement using the specified MONITOR-NAME or filename. Otherwise, no JCL changes are required.

Changes might be required to the VSE IDSYSFT (batch file table). Five types of sequential file dataviews are supported under VSE: Disk, standard-label tape, non-labeled tape, printer, and punch. One entry exists in the CA Ideal batch file table (IDSYSFT) for each type as shown in the following table:

File Dataviews	**Batch File Table Entry**
Disk	IDISK1
Standard labeled tape	ISLTAPE1
Unlabeled tape	INLTAPE1
Printer	IPRT1
Punch	IPUNCH1

You must add more entries to the file table if a program is run in batch with more than one of the same type of file. To add additional entries, change the name by incrementing the digit at the end of the file name (from 1 to G) without leaving gaps. Each type has its own set of one or more ROSFD entries.

The following are the installed ROSFD entries for sequential file dataviews:

```
IDISK1 ROSFD ACCMETH=SEQ,   for disk        X
  DTFTYPE=DTFSD,                   X
  DTFNAME=IDISK1, overridden at open        X
  DEVADDR=SYS000, overridden in JCL         X
  RECFM=FB,                     X
  DEVICE=nnnn,   installation option        X
  LRECL=4096,   overridden at open        X
  IBLKSZ=4096,   max overridden at open      X
  OBLKSZ=4104,   max overridden at open      X
  OPSYS=VSE
ISLTAPE1 ROSFD ACCMETH=SEQ, for standard-label tape  X
  DTFTYPE=DTFMT,                   X
  DTFNAME=SLTAPE, overridden at open        X
  DEVADDR=SYSnnn, opt overridden at open      X
  FILABL=STD,                    X
  RECFM=FB,                     X
  LRECL=4096,   overridden at open        X
  BLKSIZE=4096,  max overridden at open      X
  OPSYS=VSE
INLTAPE1 ROSFD ACCMETH=SEQ, for no-label tape     X
  DTFTYPE=DTFMT,                   X
  DTFNAME=NLTAPE, ignored              X
  DEVADDR=SYSnnn, opt overridden at open      X
  FILABL=NO,                     X
  RECFM=FB,                     X
  LRECL=4096,   overridden at open        X
  BLKSIZE=4096,  max overridden at open      X
  OPSYS=VSE

IPRT1 ROSFD ACCMETH=SEQ,    for a printer     X
  DTFTYPE=DTFPR,                   X
  DEVICE=1403,                    X
  DEVADDR=SYSnnn, opt overridden at open      X
  CTLCHR=ASA,                    X
  RECFM=F,                      X
  BLKSIZE=133,   max overridden at open      X
  OPSYS=VSE
IPUNCH1 ROSFD ACCMETH=SEQ,   for a punch      X
  DTFTYPE=DTFCDP,                  X
  DEVICE=2540,                    X
  DEVADDR=SYSPCH, opt overridden at open      X
  RECFM=F,                      X
  BLKSIZE=80,   max overridden at open      X
  OPSYS=VSE
```

Removing entities from the Batch CA Ideal File Table that your site never uses saves memory use. These entries provide an outline of the information needed at run time.

Before the Batch CA Ideal File Table is opened, additional or overriding information is stored in the DTF. Information can be extracted from the dictionary or the dataview definition. You can override some of this information by using the ALTER PROGRAM and ASSIGN DATAVIEW commands described in the *Command Reference* Guide. This information, with information from the CA Ideal File Table entry, provides the information needed to access the file.

Device Type

Indicates whether a sequential file dataview is disk (DISK), standard-label tape (SLTAPE), non-labeled tape (NLTAPE), printer (PRT), or punch (PUNCH).

For dataviews modeled in the dictionary, this is obtained from the dictionary when the dataview is cataloged.

For *unmodeled dataviews*, this is obtained from the dataview parameters definition.

At run time, the device type specifies to CA Ideal what type of file table entry to use: DISK, SLTAPE, NLTAPE, PRT, or PUNCH.

You can override this entry with an ALTER PROGRAM or ASSIGN DATAVIEW command before the run. For a description of the commands, see the *Command Reference Guide*.

Record Size

Indicates logical record length.

For *dataviews modeled in the dictionary*, this information is obtained from the dictionary when the dataview is cataloged.

For *unmodeled dataviews*, this information is obtained from the dataview parameters definition.

It is stored into the ROSFD entry before the file is opened.

The *Administration Guide* describes how to update the IDSYSFT.

Block Size

Indicates the physical block size.

For *dataviews modeled in the dictionary*, this information is obtained from the dictionary when the dataview is cataloged.

For *unmodeled dataviews*, this information is obtained from the dataview parameters definition.

Filename

Indicates the DLBL name for disk files or the DESTID for use in CICS.

For *modeled dataviews*, this information comes from the MONITOR-NAME attribute of the dictionary FILE entity at the time a dataview is cataloged.

For *unmodeled dataviews*, this information is obtained from the dataview parameters definition.

Logical Unit Assignment

For standard-label tape, non-labeled tape, printer, and punch files, the default assignment comes from the CA Ideal batch file table (IDSYSFT) entry. For more information, see the *Administration Guide.*

You can override the logical unit assignment with an ASSIGN DATAVIEW command before the RUN command or an ASSIGN statement executed before the first FOR construct that references the dataview. For more information, see the *Programming Reference Guide*.

The JCL does the complete logical assignments for disk files.

Modifying Dataview Characteristics Under VSE

You can modify certain environmental characteristics of dataviews without changing the dictionary file model, recataloging, and recompiling.

Under VSE, when an application program is compiled, the dataviews specified as resources are compiled into the object code, including the block size and device type. To change the block size or device type, use the ALTER PROGRAM command. This permanently alters the record length, block size, or device type.

To temporarily alter the block size, device type, or logical unit assignment for any subsequent runs in the current session, use the ASSIGN DATAVIEW command.

The RESET DATAVIEW command can reassign an assigned dataview to the database it referenced when the program was compiled. This command resets the database ID, block size, device type, and logical unit assignment.

For more information about description and syntax of the commands, see the *Command Reference Guide*.

Chapter 5: Creating Dataviews for VSAM Files

This chapter describes how to create and maintain CA Ideal dataview definitions for VSAM files. Dataviews for CA Datacom/DB/DB CBS access, SQL access, and sequential files are described in other chapters of this guide.

Note: To create and maintain CA Ideal dataview definitions for VSAM files, your site must have the CA Ideal VSAM support option installed.

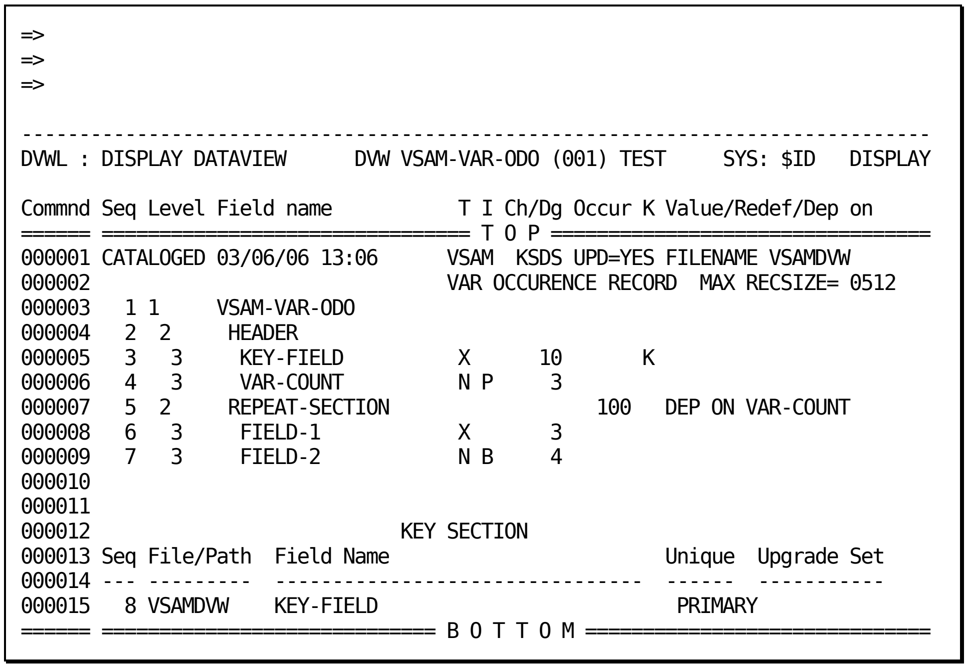
CA Ideal supports dataviews for all three types of VSAM files:

- **KSDS** Key Sequenced Data Sets
- **ESDS** Entry-Sequenced Data Sets
- **RRDS** Relative Record Data Sets

  The record length in a VSAM file can vary in two ways:

- **Variable-occurrence records** include a field or group of fields that repeats a specified number of times. These records are defined using an OCCURS DEPENDING ON clause to indicate which field controls the number of times the repeating field or group of fields occurs in the individual record. A variable-occurrence record cannot also contain redefined fields that change the length of the original field.

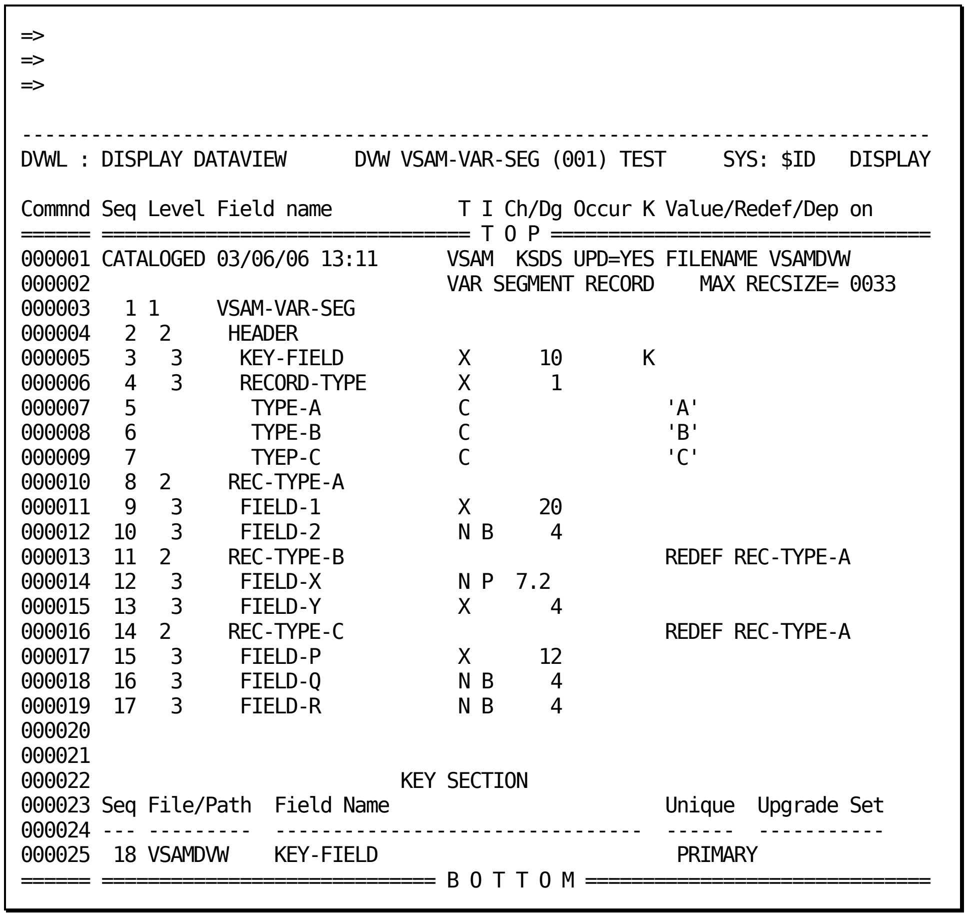
- The following screen displays the variable-occurrence records:



```
=>
=>
=>

------------------------------------------------------------------------------
DVWL : DISPLAY DATAVIEW      DVW VSAM-VAR-ODO (001) TEST     SYS: $ID   DISPLAY

Commnd Seq Level Field name           T I Ch/Dg Occur K Value/Redef/Dep on
====== ================================ T O P ================================
000001 CATALOGED 03/06/06 13:06      VSAM  KSDS UPD=YES FILENAME VSAMDVW
000002                               VAR OCCURENCE RECORD  MAX RECSIZE= 0512
000003   1 1     VSAM-VAR-ODO
000004   2  2     HEADER
000005   3   3     KEY-FIELD          X      10       K
000006   4   3     VAR-COUNT          N P     3
000007   5  2     REPEAT-SECTION                  100   DEP ON VAR-COUNT
000008   6   3     FIELD-1            X       3
000009   7   3     FIELD-2            N B     4
000010
000011
000012                              KEY SECTION
000013 Seq File/Path  Field Name                        Unique  Upgrade Set
000014 --- ---------  -------------------------------  ------  -----------
000015   8 VSAMDVW    KEY-FIELD                          PRIMARY
====== ============================= B O T T O M ==============================
```

Variable-segment records include different fields, based on a record type that is usually included in a fixed portion of the record. The varying sets of fields (segments) are defined using the REDEFINES clause to establish different level-2 field descriptions that overlap each other. These varying segments can be different lengths, but the longest segment must be the first variable segment defined. A variable-segment record cannot also contain variably repeating fields.

- The following screen displays the variable-segment records:



```
=>
=>
=>

-------------------------------------------------------------------------------
DVWL : DISPLAY DATAVIEW      DVW VSAM-VAR-SEG (001) TEST     SYS: $ID   DISPLAY

Commnd Seq Level Field name           T I Ch/Dg Occur K Value/Redef/Dep on
====== ================================ T O P ================================
000001 CATALOGED 03/06/06 13:11      VSAM  KSDS UPD=YES FILENAME VSAMDVW
000002                               VAR SEGMENT RECORD    MAX RECSIZE= 0033
000003   1 1     VSAM-VAR-SEG
000004   2  2     HEADER
000005   3   3     KEY-FIELD          X      10       K
000006   4   3     RECORD-TYPE        X       1
000007   5          TYPE-A            C                 'A'
000008   6          TYPE-B            C                 'B'
000009   7          TYEP-C            C                 'C'
000010   8  2     REC-TYPE-A
000011   9   3     FIELD-1            X      20
000012  10   3     FIELD-2            N B     4
000013  11  2     REC-TYPE-B                            REDEF REC-TYPE-A
000014  12   3     FIELD-X            N P  7.2
000015  13   3     FIELD-Y            X       4
000016  14  2     REC-TYPE-C                            REDEF REC-TYPE-A
000017  15   3     FIELD-P            X      12
000018  16   3     FIELD-Q            N B     4
000019  17   3     FIELD-R            N B     4
000020
000021
000022                           KEY SECTION
000023 Seq File/Path  Field Name                        Unique  Upgrade Set
000024 --- ---------  --------------------------------  ------  -----------
000025  18 VSAMDVW    KEY-FIELD                          PRIMARY
====== ============================= B O T T O M ==============================
```

KSDS files require a fixed-length segment, since the keys must always be in the same position in the record.

ESDS files are not required to have a fixed-length segment. If an ESDS file does not have a fixed-length segment, you can determine the record type from the length using the function $REC-LENGTH. For more information about $REC-LENGTH function, see the *Command Reference Guide*.

For RRDS files, variable-occurrence records are not supported (by VSAM). Variable-segment records are accommodated to allow the use of multiple record types; however, the records are padded with binary zeros, as necessary, to create fixed-length records.

VSAM dataviews are created and defined in CA Ideal. They are not modeled in the dictionary facility.

Defining VSAM Dataviews

CA Ideal provides a facility for creating, describing, and maintaining dataview definitions for VSAM files. A VSAM dataview is recorded as a DATAVIEW entity occurrence in the dictionary. The definition of the fields comprising the dataview is recorded in a VLS member. In addition, the primary and alternate index keys for the VSAM file are also recorded in the VLS member.

Like the unmodeled sequential file dataview, the VSAM dataview is created in CA Ideal and can be edited, duplicated to a new version or a new name, and deleted in CA Ideal. Multiple versions of the dataview can be maintained. They can be in TEST, PROD, or HIST status.

The dataview definition must be cataloged. You can display and print it using CA Ideal commands.

Components of a VSAM Dataview Definition

To define the components of a CA Ideal VSAM dataview, use the following fill-in panels:

- An *Identification* fill-in creates a VSAM dataview definition and provides descriptive information about it. The section titled Dataview Identification Fill-In that follows explains how to complete this fill-in.
- A *Parameter Definition* fill-in describes the file associated with the dataview. The section Dataview Parameter Definition in this chapter describes how to complete this fill-in.
- A *Field Definition* fill-in describes the fields comprising the dataview. The section Dataview Field Definition in this chapter describes how to complete this fill-in.
- A *Key Definition* fill-in describes the primary and alternate indexes that access a VSAM KSDS data set. This fill-in is not used for RRDS and ESDS data sets. The section Dataview Key Definition in this chapter describes how to complete this fill-in.

How to Create a VSAM Dataview Definition

The first step in creating a VSAM file dataview is identifying the dataview. To identify the dataview, you must first enter the CREATE DATAVIEW command, and then fill in the Dataview Identification screen. Once you name the dataview in the CREATE DATAVIEW command or on the Identification fill-in, an entry for the dataview definition is stored in the dictionary facility. After the dataview is identified, it becomes the current definition.

CREATE DATAVIEW creates version one, in test status. You can modify, print, duplicate, delete, and mark it to production status. A dataview definition does not have to be in production status before it can be cataloged.

Dataview Identification Fillin

The Dataview Identification fill-in in the following figure enters descriptive information about the VSAM dataview, either when the dataview definition is initially created or when it is subsequently modified.

To display the Dataview Identification fill-in:

- For a new dataview definition, enter the command CREATE DATAVIEW.
- For an existing dataview definition, enter the EDIT or DISPLAY DATAVIEW command, specifying the IDENTIFICATION component in the command:

  `EDIT DVW` *`name`* `IDE`

- For the current dataview, enter the IDENTIFICATION command or press the equivalent PF key.

```
=>
=>
=>
-------------------------------------------------------------------------------
 IDEAL: DVW IDENTIFICATION    DVW                            SYS: CCF   FILL-IN

DATAVIEW Name     __________________

Access-Method     ____ (SEQ|VSAM)

Operating System MVS  (MVS|VSE)



Short Description ________________________
Description:
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
```

The following data displays on the dataview identification fill-in:

Dataview Name

Enter the 1- to 18-character name assigned to the VSAM dataview definition. If you enter a name in the CREATE command or prompter, CA Ideal initializes the name to the supplied value.

Access Method

Enter the type of file defined: VSAM for a VSAM file or SEQ for a sequential file. Once you enter the Access Method and it is accepted, you cannot modify this value.

Operating System

Enter the type of operating system under which the dataview is accessed. The default value for this field is the current system.

Created... By

Indicates the initial creation date of the dataview definition and the user ID of the creator. This information appears only after the name is entered and accepted.

Last Modified ...at ...By

Indicates the date, time, and user ID of the last edit access. This is blank until an EDIT DATAVIEW command accesses this definition. This information appears only after the dataview name is entered and accepted.

Last Cataloged ...at ...By

Indicates the date, time, and user ID when this dataview was last cataloged. This information appears only after the dataview name is entered and accepted. The date and time are set to zeros when the dataview is created, and are reset when the dataview is cataloged.

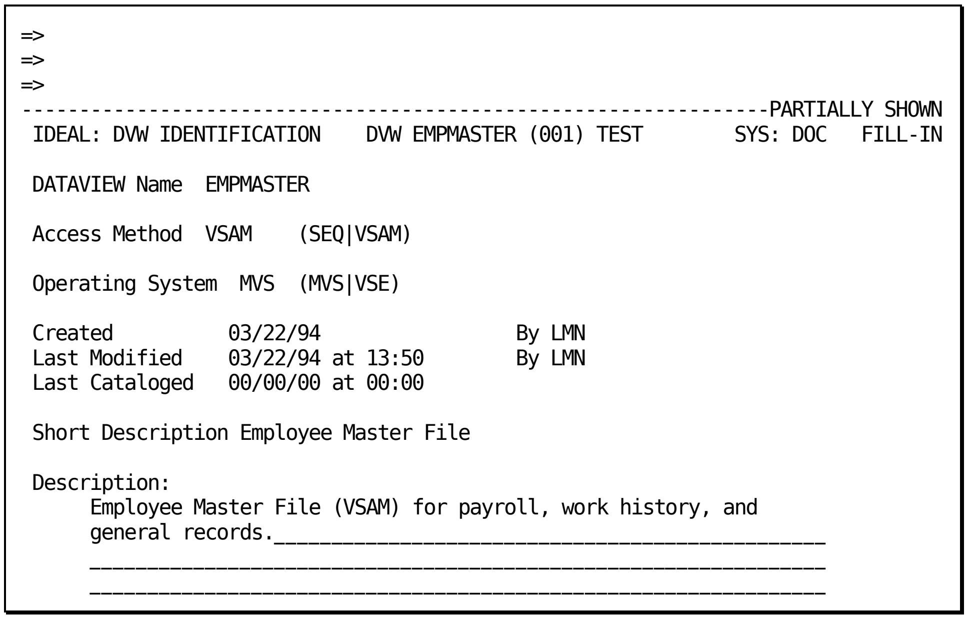
Short Description

Enter a brief description of the dataview definition. This field does not require an entry. The short description displays in the list the DISPLAY INDEX DATAVIEW command provides.

Description

Enter a full description of the dataview definition, up to five lines long. This field does not require an entry.

The following screen displays the dataview identification fill-in:



```
=>
=>
=>
----------------------------------------------------------------PARTIALLY SHOWN
 IDEAL: DVW IDENTIFICATION    DVW EMPMASTER (001) TEST        SYS: DOC   FILL-IN

 DATAVIEW Name  EMPMASTER

 Access Method  VSAM    (SEQ|VSAM)

 Operating System  MVS  (MVS|VSE)

 Created         03/22/94                 By LMN
 Last Modified   03/22/94 at 13:50        By LMN
 Last Cataloged  00/00/00 at 00:00

 Short Description Employee Master File

 Description:
      Employee Master File (VSAM) for payroll, work history, and
      general records.________________________________________________
      ________________________________________________________________
      ________________________________________________________________
```

Dataview Parameter Definition

The dataview parameter definition describes the VSAM file corresponding to the dataview. Parameter data is specified on the following Dataview Parameter fill-in:

```
=>
=>
=>
--------------------------------------------------------------------------------
 IDEAL: DVW PARAMETERS        DVW EMPMASTER (001) TEST        SYS: DOC   FILL-IN

File Organization   KSDS       (KSDS|ESDS|RRDS)
Update Intent       Y          (Y=Yes, N=No)
Filename            VSAMDVW
```

The fields on the dataview parameter fill-in are as follows:

File Organization Enter one of the following:

KSDS-Key-sequenced data set

ESDS-Entry-sequenced data set

RRDS-Relative record data set

The default is KSDS. If you enter a value other than one of the three above, the last valid value in the field is used.

Update Intent Enter one of the following:

Y-The default, updates the data set using this dataview.

N-Prevents the data set from being updated using this dataview.

Filename Enter one of the following:

z/OS-The ddname of the base cluster

VSE-DLBL file name of the base cluster

Depending on the Operating System entry on the Identification fill-in, you can enter up to eight characters for a z/OS file name and up to seven characters for a VSE filename.

If you do not supply a file name, the name defaults to VSAMDVW.

Note: You must be sure that the CA Ideal startup JCL contains a DD or DLBL statement for each VSAM file used in CA Ideal. For CICS, you must also be sure that an FCT entry that corresponds to the filename of the base cluster is included for each VSAM file.

Dataview Field Definition

The FIELD command or equivalent PF key displays the following Dataview Field Definition fill-in for the current dataview definition. The Dataview Field Definition fill-in names and describes the fields in the dataview.

```
=>
=>
=>
-------------------------------------------------------------------------------
 IDEAL: DVW FIELD DEFINITION  DVW EMPMASTER (001) TEST       SYS: DOC   FILL-IN

Command Level Field Name      T I Ch/Dg Occur Value/Comments/Clauses
------  ----- -------------- - - ----- ---------------------------------------
======  ===== ======= TOP === = = ===== ===== =================================
......
......
......
======  ===== ===== BOTTOM == = = ===== ===== =================================
```

The fields on the Dataview Field Definition fill-in are as follows:

Command

Enter any line commands in this area. For a description of line commands, see the *Command Reference Guid*e.

Level

Enter the number from 2 to 16 that identify the hierarchical rank of the field. Since the dataview is treated as the level-1 item, field levels start with level 2. Level-2 elementary fields (fields with no subordinate fields) do not require a level number. If no level number is supplied, the level number defaults to 2.

Field Name

Enter one of the following:

- A valid name (see the *Command Reference Guide*) that identifies an elementary or group field. An elementary field is a field that has no subordinate fields. Elementary fields are further defined by values entered in the T and Ch/Dg field. Group field name entries have no T and Ch/Dg values.

  You can continue the field name on a second line by including a semicolon (;) as the last character on the first line. You can break the field name at any character. You can specify the level number and the attributes T, I, Ch/Dg, and Occur on the first line only.

  The field name of an elementary field must be unique in the dataview.

- A condition name that identifies a value for a condition type (C) field. When the Field Name is a condition name, the columns Level, I, Ch/Dg, and Occur must be blank, but Value/Comments/Clauses must contain a value.

- A valid subscript, of the form (*n*), (*n,o*), or (*n,o,p*), that identifies a specific occurrence of an occurring field. The initial value for that occurrence of the field is entered in the Value/Comments/Clauses column. The columns Level, T, I, Ch/Dg, and Occur must be blank.

Note: The depth of subscripting specified must correspond to the depth of the subscripted field. For example, a three-level table must have three subscript entries (n,o,p) to fully identify the occurrence.

- The word FILLER or blanks to indicate an unnamed non-addressable field that reserves space. If the Field Name column is blank and there are values in the T and Ch/Dg columns, the suffix FILLER is assigned to the Field Name when the dataview is cataloged.

- Blanks when the Value/Comment/Clauses column contains a continuation line, a comment, or when the rest of the line is blank.

T (Type)

Enter one of the following field types:

- **X** (Alphanumeric field)-An alphanumeric field can contain any alphabetic, numeric, or special character.
- **N** (Signed numeric field)-A signed numeric field can contain the digits 0-9, a minus sign, and a decimal symbol.
- **U** (Unsigned numeric field)-An unsigned numeric field can contain the digits 0-9 and a decimal symbol.
- **D** (Date field)-A date field contains a signed, integer number that expresses the date as the number of days before (-) or since (+) December 31, 1900 (day zero).
- **C** (Condition name assigned to a specific value of a field)-The original field name has a Type of X, and the condition names have a Type of C.

Note: The type defaults to N if the internal representation is specified but no Type is specified. The type must be blank for a group field.

The *Programming Reference Guide* contains a detailed explanation of each field type and its characteristics.

I (Internal Representation)

Enter one of the following to indicate the internal representation of numeric (signed and unsigned) and date type fields:

- **P**-Packed (This is the default.)
- **Z**-Zoned
- **B**-Binary

  Note: Internal representation must be blank unless the Type is N, U, or D.

CH/DG (Characters/Digits)

Enter the length, in characters or digits, of the field value:

- **For Type X**-Enter the number of characters allowed in the field. The maximum length for an alphanumeric field is 32,767.
- **For Type N or U**-Enter the number of integer places and the number of decimal places in the field, separated by a decimal point. The maximum length for a packed or zoned numeric field is 31. The maximum length for a binary field is 9.
- **For Type D**-Enter the number of integer places in the field. A date field cannot have decimal places. The default length for date fields is 7. The minimum length for date fields is 5. The maximum length is 31 for packed or zoned fields, 9 for binary fields.

  You must enter a value in the Ch/Dg column for all elementary field types except dates and flags. A non-date entry with no type or length information is assumed to be a group name and must have subordinate fields following it.

  In the following table, 42 in the Ch/Dg column for the first field (Type X), indicates a length of 42 characters. For the next field (Type N), the 7 in the Ch/Dg column indicates a seven-digit field with seven integer positions; and for the third field (also Type N), the 10.3 in the Ch/Dg column indicates a 13-digit field with ten integer positions and three decimal places. The last field (Type D), with a 5 in the Ch/Dg column, can hold an internal 5-integer date value of up to 273 years from the base year.

```
Type   CH/DG
X      42
N       7
N      10.3
D      5
```

Occur (Number of Occurrences)

If a field or group occurs multiple times, enter the number of times the field or group occurs. If there is a DEP ON clause in the Value/Comments/Clauses column, enter the maximum number of times the field can occur. The field named in the DEP ON clause indicates the exact number of times the field occurs for this record.

Value/Comments/Clauses

Depending on the circumstances, specify one of the following:

- Value for the occurrence of the field.
- Descriptive comment about the field. You can specify a comment alone or with a value, a redefinition keyword, or a depending on clause.
- REDEFINITION keyword.
- DEPENDING ON clause.

Note: CA Ideal retains or adjusts the case of the data entered in this column depending on the setting for case (determined either with a SET EDIT CASE command or by default).

Value

Default initial value assigned to a field. The specified value is inserted when a record is created using this dataview and no value is provided for this field. Statements in the scope of the FOR NEW construct can overwrite the default initial value.

Elementary Field

An elementary field can be assigned a value in the Value/Comments/Clauses column. This value can be a numeric or alphanumeric literal, depending on the type of the elementary field. You must enclose alphanumeric literals in delimiters (“or "). If an alphanumeric literal is longer than the space provided, continue the alphanumeric literal on the next line of the same column surrounded by a new pair of delimiters. Leave all other columns on the continuation line blank. You can also continue Field Names. Two or more alphanumeric literals continued in this fashion are concatenated and treated as one.

If a value is not entered in the Value/Comments/ Clauses column, the default value for a numeric field is zero and for an alphanumeric field, spaces.

Occurring Field

In an occurring field, the Value column is left blank on the line that contains the field name for the parent field and the following lines in the Value column can contain entries for the value of each occurrence of the field. These values are identified by a valid subscript number (of the form (n), (n,o), or (*n,o,p*)) on the corresponding line in the Field Name column.

You cannot specify initial values for group fields.

Date Field

You cannot assign an initial value for a date field. (The default initial value is 0.) This also means that you cannot define a date field with subordinate condition names or flags. However, a date field can redefine a numeric field with an initial value.

Comments

Provide useful information about the field. A comment is indicated in this column by a preceding colon (:). To continue a comment over multiple lines, start each line of the column with a colon. The other columns can be blank or the Field name column can contain the continuation of the field name.

Redefinition

The keyword REDEFINITION (or REDEF) in this column indicates that this item is another view of the closest previously defined item at the same level that is not itself a redefinition. This item cannot be larger than the original item. For an RRDS file dataview, if a redefined field has a shorter length than the original field, the record is padded with binary zeros since RRDS files must have fixed-length records.

The two items can be different types (such as alphanumeric and numeric). If a field with a REDEF is a subordinate field, its Type does not affect whether the group is alpha or non-alpha. CA Ideal uses the Type of the original item. For more information about alpha and non-alpha group fields, see the Programming Reference Guide.

An item with a REDEF cannot have an initial value. If a group has a REDEF, none of its subordinate fields can have initial values. (However, the original item can have an initial value.) An OCCURs clause and a REDEFines clause cannot be on the same level-2 field for a VSAM dataview.

To define a variable-segment record, define the varying segments as level-2 group fields. For each varying segment after the first, use the REDEFINITION keyword on the level-2 group field that names the segment to indicate that the group field redefines the first variable segment. For variable-segment records, there must be no level-2 fields after the last redefined field.

If there are redefined fields at the end of a dataview but those redefinitions do not change the record length, the file is not a variable-segment file, and the $REC-SEGMENT function cannot change the record length.

Depending on

A DEPENDING ON (DEP ON) clause in this column indicates that the field (elementary or group) repeats a variable number of times. The repeating field must be the last field at that level in the record. The field specified in the DEPENDING ON clause indicates the number of times the field named in the Field name column occurs in the current record. The DEPENDING ON field must be a non-repeating, elementary, numeric field that appears previously in the dataview and contains an integer value.

Note: An RRDS dataview cannot include variable-occurrence records.

The following example illustrates the various types of data structures that you can define using the dataview field fill-in.

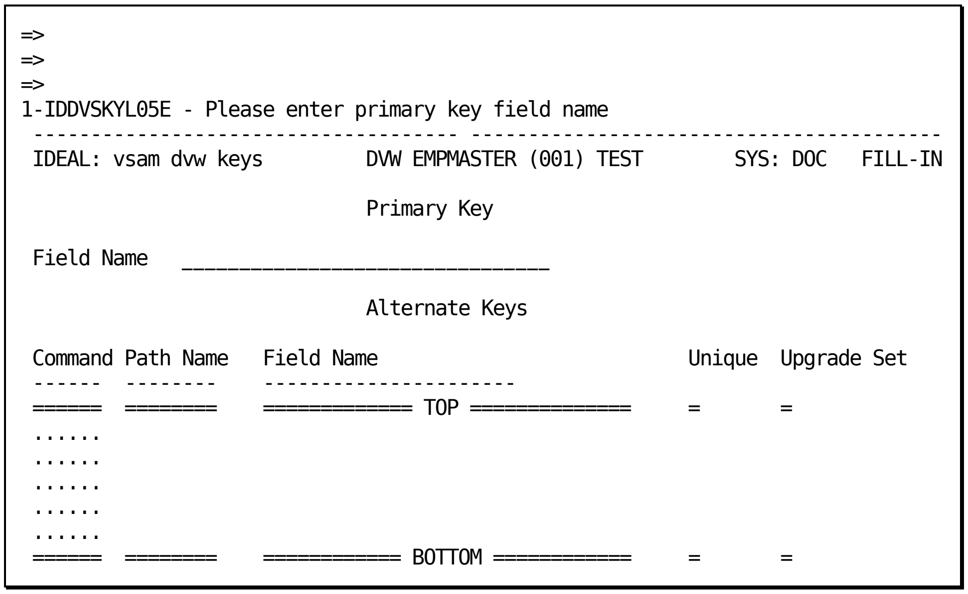
```
=>
------------------------------------------------------------------------------
IDEAL: DVW FIELD DEFINITION  DVW EMPMASTER (001) TEST        SYS: DOC   FILL-IN

Command Level Field Name          T I Ch/Dg Occur Value/Comments/Clauses
------  -----  ------------- - - ----- --------------------------------------
======  ===== ======= TOP ======= = = ===== ===== ============================
000100   2    NAME
000200    3   LAST-NAME           X      20
000300    3   FIRST-NAME          X      15
000400    3   MID-INIT            X       1
000500   2    EMP-ID              N Z     7
000600   2    ADDRESS
000700    3   STREET              X      20
000800    3   CITY                X      15
000900    3   STATE               X       2
001000    3   ZIP-CODE            X       9
001100    3   ALT-ZIP             N Z     9       REDEF
001200   2    MARITAL-STATUS      X       1
001300        MARRIED             C               'M'
001400        SINGLE              C               'S'
001500        DIVORCED            C               'D'
001600        WIDOWED             C               'W'
001700        SEPARATED           C               'E'
001800   2    DEPENDENTS
001900    3   NUM-DEP             N Z     2       :NUMBER OF DEPENDENTS
002000    3   DEP-DATA                         12 DEP ON NUM-DEP
002100     4  DEP-NAME
002200      5 GIVEN               X      12       :DEPENDENT'S FIRST NAME
002300      5 MIDDLE              X       1       :            MIDDLE INITIAL
002400      5 LAST                X      20       :            LAST NAME
002500                                            :IF DIFFERENT FROM EMPLOYEE
002600     4  BIRTH-DATE          D Z     8
002700     4  SEX                 X       1
======  ===== ===== BOTTOM ====== = = ===== ===== ============================
```

Dataview Key Definition

The dataview key definition specifies the primary and alternate keys (indexes) that can access records through this dataview. You must complete the key definition for KSDS files. You cannot define keys for RRDS and ESDS files. (CA Ideal does not support alternate indexes for ESDS or RRDS files.)

Note: If the file contains variable-occurrence or variable-segment records, the keys must all be in the fixed-length portion of the record.

Enter the KEY command to display the Dataview Key Definition fill-in shown below.



```
=>
=>
=>
1-IDDVSKYL05E - Please enter primary key field name
 ------------------------------------ ----------------------------------------
 IDEAL: vsam dvw keys         DVW EMPMASTER (001) TEST        SYS: DOC   FILL-IN

                              Primary Key

 Field Name   ________________________________

                              Alternate Keys

 Command Path Name   Field Name                         Unique  Upgrade Set
 ------  --------    ----------------------
 ======  ========    ============= TOP ==============     =       =
 ......
 ......
 ......
 ......
 ......
 ======  ========    ============ BOTTOM ============     =       =
```

The fields on the Dataview Key Definition fill-in are:

Primary Key Field Name

Enter the name of the field that is the complete primary key. The key must be in the fixed portion of a variable-occurrence or variable-segment record. If the key is composed of several fields, a group field must be defined for the key. The key field can be a redefined field, but it cannot be a field that is part of a repeating group field (referenced with a subscript) nor can it be a condition name.

An entry is required for this field.

Alternate Keys

Definition of alternate keys is optional. It is not necessary to include all or any of the alternate indexes defined for the VSAM file in your dataview definition. To define an alternate key, use the following fields:

Command

Enter CA Ideal line commands in this area, as needed. For a description of the CA Ideal line commands, see the *Command Reference* Guide.

Path Name

Enter the ddname or DLBL file name of the JCL statement that corresponds to the path established with IDCAMS for the alternate index. The Path Name can be up to eight characters long in a z/OS environment or up to seven characters long in a VSE environment. The path name must be unique in the path and file names specified for the dataview.

Field Name

Enter the field name that identifies the full alternate key. The key must be in the fixed portion of a variable-occurrence or variable-segment record. If the key is composed of several fields, a group field must be defined for the key. The key field can be a redefined field, but it cannot be a field that is part of a repeating group field (referenced with a subscript) or a condition name.

An alternate key can overlap the primary key or another alternate key fully or partially. However, the field name must be unique for each key. This means that you must redefine the field to provide a unique name for each key.

Unique

Enter one of the following:

- **Y-**Key is a unique key. This is the default.
- **N-**Key is a non-unique key.

If the value in this field does not correspond to the actual attribute of the alternate index as it is defined with IDCAMS for the given path, unpredictable results occur.

The execution of a FOR EACH statement based on a non-unique key cannot resume after any of the following events takes place in the FOR construct:

- Panel TRANSMIT
- CHECKPOINT
- BACKOUT
- Debugger breakpoint
- Update of the current data set, even if the update was made through a different dataview

Upgrade Set

Enter one of the following:

Y-This key is part of the upgrade set defined for the VSAM cluster with IDCAMS. If this is specified, a DD or DLBL statement must be present for this path and, in CICS, an FCT entry must also be present for the specified Path Name.

N-This key is not part of the upgrade set. This is the default.

If the dataview is updateable (Update Intent = Y on the Parameter Definition fill-in), all paths that are defined as part of the upgrade set are validated at runtime, regardless of which one accesses the data set.

Note: This attribute must match the VSAM catalog definition for the path.

In the following example, both a primary key and one alternate key were defined.

```
=>
=>
=>
-------------------------------------------------------------------------
 IDEAL: vsam dvw keys         DVW EMPMASTER (001) TEST        SYS: DOC  FILL-IN

                              Primary Key

 Field Name   NAME

                              Alternate Keys

 Command Path Name   Field Name                          Unique  Upgrade Set
 ------  --------    --------------------------------
 ======  ========    ============= TOP =============     =       =
 000100  EMPID       EMP-ID                              Y       Y
 ======  ========    ============ BOTTOM ============     =       =
```

Note: VSAM implementation in CA Ideal requires that the master file and all alternate indexes included in the Upgrade Set be defined with SHAREOPTIONS (2,3) or higher.

Runtime Considerations for VSAM Dataviews

When executing a CA Ideal program with a VSAM dataview, the considerations covered in this section apply.

Validating the Dataview at Runtime

The first time a VSAM dataview is accessed during the execution of a program, the dataview is validated for the following attributes:

- File organization (ESDS, KSDS, or RRDS)
- Maximum record length (determined by totaling the lengths of the fields defined in the Field Definition fill-in)
- Displacement and length of each primary and alternate key defined in the Dataview Key Definition fill-in

The source against which the attributes of a VSAM file are validated depends on the runtime environment. In CICS, the FCT entry for the base cluster and alternate path are used. The FCT entry for variable-occurrence or variable-segment files must indicate that the file has records of varying lengths. In all other environments, the information is obtained by opening the relevant data sets.

The extent of validation depends on the update attribute of the dataview. If the dataview is updateable in the current program, the whole upgrade set defined for the dataview is validated (for KSDS data sets). If the dataview can only be read (as defined in the program, in the dataview definition, or at runtime), only the current access path is validated. Index uniqueness and membership in the upgrade set are not validated.

If validation fails, a runtime error results.

Updating: Use of the Primary Index

Although CA Ideal can retrieve records from a VSAM file using either the primary or an alternate key (index), CA Ideal only updates records in a VSAM file using the primary index. This means that a record that was read and must be updated or deleted is reread with exclusive control using the primary index. CA Ideal then verifies that the record was not deleted or changed since the original access before updating or deleting the record.

Restrictions on NonUnique Keys

The following restrictions apply to the use of a non-unique key:

- TRANSMIT statements and debugger breakpoints result in CICS syncpoints. They are not restricted in any way if the access key is unique. For a non-unique key, the TRANSMIT or breakpoint occurs, but the execution of a FOR EACH/FIRST statement cannot resume thereafter. Any attempt to resume execution under such conditions terminates the run with an error message.
- Use of the CHECKPOINT and BACKOUT statements is subject to the same restrictions on the resumption of a FOR EACH/FIRST statement using a non-unique key.
- The execution of a FOR EACH statement based on a non-unique key for a data set that was updated cannot resume following the update or deletion. This restriction affects all nested dataviews on the same data set. If an attempt is made to resume execution under such conditions, an error message is issued, and the run terminates.

CHECKPOINT and BACKOUT With VSAM Files

In the CICS environment, the CHECKPOINT and BACKOUT statements execute the CICS SYNCPOINT and SYNCPOINT ROLLBACK commands. For VSAM data sets, CA Ideal takes no further actions. CICS handles these processes exclusively.

In a non-CICS environment, the CHECKPOINT statement executes a TCLOSE operation that flushes certain VSAM buffers if there was any VSAM access before the CHECKPOINT. The BACKOUT statement has no effect for VSAM files. This can lead to inconsistency between a CICS and non-CICS execution of the same program.

In CICS, a lock is held on updateable records until a CHECKPOINT is issued. In batch, a lock is held on only one record at a time. This means that in batch, VSAM files are not recoverable.

Opening and Closing VSAM Files

A VSAM file is opened the first time a FOR construct is encountered that uses the specified access path. The base cluster is opened with update intent unless No Update was specified in the dataview definition, in the FOR construct, or at runtime. The file is not normally closed until the end of the run. However, if the base cluster is opened for read-only access and a subsequent FOR construct allows updating, the file is closed and re-opened with update intent.

VSAM data sets are protected by VSAM and, in the CICS environment, CICS. CA Ideal does not provide any other sharing or locking mechanisms.

Preparing for Execution

Before you can use a VSAM file in a CA Ideal program, you must identify the file to the operating system and to CA Ideal. You identify the file to CA Ideal by creating the dataview. To identify the file to the operating system, follow the appropriate steps, below.

In CICS

Define the file using RDO.

In Batch

Add a DD statement (z/OS) or DLBL statement (VSE) to the CA Ideal batch JCL.