CA IDMS - 19.0

Using ADS Batch

Date: 13-Jan-2018
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

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

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

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

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

The manufacturer of this Documentation is CA.

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

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

## Introduction to ADS Batch

- Example 1 Employee-record Archiving Application ................................................................. 9
- Example 2 Employee-record Restore Application ................................................................. 10

## CA ADS Batch Concepts

- Input and Output Files and File Maps ..................................................................................... 12
- Suspense Files .......................................................................................................................... 15
- Log Files .................................................................................................................................... 16
  - Log File Information .................................................................................................................. 17
  - Log File Prefix .......................................................................................................................... 18
- Archiving .................................................................................................................................. 18
- Batch Dialog Structure ............................................................................................................. 18
  - CA ADS Batch Dialog Components ....................................................................................... 19
  - Batch Control Events ............................................................................................................. 20
  - Batch Response Field Values ............................................................................................... 20
  - Batch Dialog Execution ......................................................................................................... 21
- Application Structure ............................................................................................................... 24
  - Disallowed Functions ............................................................................................................. 25
  - Application Flow of Control .................................................................................................. 25
  - Accessing Input Files with Multiple Record Layouts .............................................................. 27

## Application Compiler

- General Options Screens ......................................................................................................... 30
- Global Records Screen ............................................................................................................ 31
- Task Codes Screen .................................................................................................................. 31
- Response/Function List Screen ............................................................................................... 32
- Response Definition Screen ................................................................................................... 33
- Function Definition Screen ..................................................................................................... 34

## Dialog Compiler

- Dialog Compiler in Online Mode ............................................................................................. 36
  - Main Menu Screen ................................................................................................................. 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Specifications Screen</td>
<td>37</td>
</tr>
<tr>
<td>Database Specifications</td>
<td>38</td>
</tr>
<tr>
<td>Options and Directives Screen</td>
<td>38</td>
</tr>
<tr>
<td>Records and Tables Screen</td>
<td>39</td>
</tr>
<tr>
<td>Process Modules Screen</td>
<td>40</td>
</tr>
<tr>
<td>Dialog Compiler in Batch Mode (ADSOBCOM)</td>
<td>41</td>
</tr>
<tr>
<td>CA IDMS/DC Mapping Facility</td>
<td>41</td>
</tr>
<tr>
<td>Batch Automatic Editing and Error Handling</td>
<td>41</td>
</tr>
<tr>
<td>Variable Array Processing</td>
<td>42</td>
</tr>
<tr>
<td>Online Mapping Facility</td>
<td>43</td>
</tr>
<tr>
<td>Map Definition Screen</td>
<td>44</td>
</tr>
<tr>
<td>Extended Map Definition Screen</td>
<td>44</td>
</tr>
<tr>
<td>File Field Selection Screen</td>
<td>45</td>
</tr>
<tr>
<td>File Field Edit Screen</td>
<td>46</td>
</tr>
<tr>
<td>Batch Map Compiler (RHDCMAP1)</td>
<td>48</td>
</tr>
<tr>
<td>ADSOBSYS</td>
<td>48</td>
</tr>
<tr>
<td>z/VSE File Characteristics Program</td>
<td>53</td>
</tr>
<tr>
<td>DEFINE CHARACTERISTICS Statement</td>
<td>55</td>
</tr>
<tr>
<td>ACCEPT CHARACTERISTICS statement</td>
<td>58</td>
</tr>
<tr>
<td>RUN PROGRAM Statement</td>
<td>58</td>
</tr>
<tr>
<td>CA ADS Batch Print Log Utility</td>
<td>59</td>
</tr>
<tr>
<td>How to Use the Print Log Utility</td>
<td>59</td>
</tr>
<tr>
<td>DSECT for the Log File Prefix</td>
<td>60</td>
</tr>
<tr>
<td>ADS Batch Control Statements</td>
<td>61</td>
</tr>
<tr>
<td>z/OS JCL for ADSOBPLG</td>
<td>63</td>
</tr>
<tr>
<td>CA ADS Batch Sample Applications</td>
<td>67</td>
</tr>
<tr>
<td>Employee-record Archive Application</td>
<td>67</td>
</tr>
<tr>
<td>Step 1 Describe the Files in the Data Dictionary</td>
<td>69</td>
</tr>
<tr>
<td>Step 2 Describe the Records in the Data Dictionary</td>
<td>69</td>
</tr>
<tr>
<td>Step 3 Define the File Maps</td>
<td>71</td>
</tr>
</tbody>
</table>
Step 4 Define the Process Modules ................................................................. 72
Step 5 Define the Dialogs .............................................................................. 72
  Dialog ARCD1 ......................................................................................... 73
  Dialog ARCD2 ......................................................................................... 76
  Dialog ARCD3 ......................................................................................... 78
  Dialog ARCD4 ......................................................................................... 80
  Dialog ARCD5 ......................................................................................... 82
Executing the Application ........................................................................... 83
Employee-Record Restore Application ......................................................... 84
  Step 1 Define the Application Structure ..................................................... 86
  Step 2 Define the Process Modules .............................................................. 87
  Step 3 Define the Dialogs ........................................................................... 88
    Dialog RESD1 ......................................................................................... 88
    Dialog RESD2 ......................................................................................... 92
    Dialog RESD3 ......................................................................................... 96
Executing the Application ........................................................................... 97
Employee-Record Report Application .......................................................... 98
  Step 1 Describe the Records in the Data Dictionary .................................... 101
  Step 2 Define the File Maps ...................................................................... 102
  Step 3 Define the Process Modules .............................................................. 103
  Step 4 Define the Dialogs ........................................................................... 104
    Dialog REPTEDIT .................................................................................... 104
    Dialog DEPTHDR .................................................................................... 108
    Dialog COLMHDR .................................................................................... 109
    Dialog DLINEOUT .................................................................................... 110
    Dialog HDRLINE ..................................................................................... 112
Executing the Application ........................................................................... 113
Using ADS Batch

CA Application Development System/Batch (CA ADS Batch) is a facility that allows you to develop fourth-generation batch applications that execute in the CA IDMS environment. CA ADS Batch applications can read data from input files, perform CA IDMS/DB database update and retrieval, and write data to output files.

Note: Do not confuse batch application execution with batch application definition. CA ADS Batch refers to Application Development System applications that execute in the batch environment. CA ADS Batch application components can be defined in online or batch mode.

Common Facilities

CA ADS Batch uses many of the same facilities as CA ADS, including:

- The **CA IDMS mapping facility**, used to define **maps** through which all transaction input and output is performed. In the CA ADS environment, map definitions control the transfer of data between a terminal operator's screen and dialog variable storage. In the CA ADS Batch environment, map definitions control the transfer of data between input and output files and variable storage. Maps provide extensive automatic editing and error handling features that simplify application development. These features, formerly available only in the online environment, are now also available in the batch environment.

- The **dialog compiler**, used to define **dialogs** that read, process, and write transactions at runtime.

- The **application compiler**, used to create a global structure of an application.

- The **runtime system**, used to execute applications.

- The **Integrated Data Dictionary** (IDD), used to define certain application components, such as process modules and file definitions, and to provide centralized documentation of all application components. The data dictionary also provides automatic cross-referencing of components, reporting facilities, and security features.

Differences

CA ADS Batch differs from CA ADS mainly in its handling of transaction input and output. Online transactions are typically mapped in and mapped out through a terminal. Batch transactions are typically read from and written to files, such as sequential data sets.

Batch Features

CA ADS Batch includes the following features in support of the batch runtime environment:
Input and output file support—CA ADS Batch provides access to input and output sequential files, such as disk, tape, card, and printer, and to VSAM entry-sequenced data sets (ESDS) that are accessed sequentially. File records can be any type except variable spanned.

Suspense file support—CA ADS Batch can write erroneous input file records to a suspense file. After an application has been executed, the user can correct the records in the suspense file, then resubmit them directly in another run of the application.

Log file support—CA ADS Batch accumulates in a system log runtime information, such as error messages and dialog statistics. CA ADS Batch provides a log file archiving utility (z/OS), which writes log information to tape when the log file is full. CA ADS Batch also provides a print log utility that prints formatted reports of selected log file information.

Process commands—CA ADS Batch uses commands to read from and write to input, output, and log files, and to send messages to the operator’s console. Environment test conditions are also included (for example, IF $ONLINE THEN DISPLAY. ..., or, IF $BATCH THEN DO. ...); thus, a process module can be used in both the online and batch environments.

Sample Applications
The remainder of this section introduces two examples of CA ADS Batch applications. These examples are illustrated in detail in **CA ADS Batch Sample Applications** (see page 67).

## Introduction to ADS Batch

### Example 1 Employee-record Archiving Application

The employee-record archiving application writes selected employee records and their associated insurance coverage records from an CA IDMS/DB database to tape.

The figure below shows the I/O performed by the application.

![Diagram showing I/O for Employee-record Archiving Application](image)

**Application Notes**

The following notes describe the application:

- Each input file record contains the id of an employee record to be archived.
- Each specified employee record is retrieved from the CA IDMS/DB database and written to an output file.
- All insurance coverage records associated with an employee being archived are retrieved from the database and written to the same output file.
Coverage records are deleted as they are archived. Employee records are deleted after all associated coverage records have been archived.

A transaction summary report file is created.

Invalid input records are written to a suspense file.

A log file accumulates error messages and other information.

Example 2 Employee-record Restore Application

The employee-record restore application restores to the database selected archived employee records and their associated coverage records.

The following figure shows the I/O performed by the employee-record restore application.

---

Application Notes

The following notes describe the application:

- One of the two input files contains the ids of employee records to be restored.

- The second input file contains the archived employee records and their associated coverage records.
Based on employee ids specified in the first input file, the application restores employee and associated coverage records from the second input file.

- A transaction summary report file is created.
- A suspense file stores all invalid records from the first input file.
- No suspense file is allocated for the second input file. It is assumed that the archived employee and coverage records are formatted correctly. Note, however, that a suspense file can be allocated for each input file in an application.
- A log file accumulates error messages and other information.

CA ADS Batch Concepts

Developing an CA ADS Batch application is, in many ways, similar to developing an online application with CA ADS. The major difference is that in CA ADS Batch you describe data transfer between variable storage and files, while in CA ADS, you describe data transfer between variable storage and online terminals.

Steps

To develop and execute an CA ADS Batch application, you perform the following steps:

1. Create external file descriptions. You describe your input and output files in the data dictionary with file, record, and element entities. You can define files by using DDDL.

2. Define file maps. You describe how data is to be transferred at runtime between the input and output files and variable storage. You can define file maps by using the online mapping facility.

3. Create process modules. You define in the data dictionary the process logic required by the application. The process command language includes new commands for batch processing. You can define process modules by using DDDL.

4. Define dialogs. You define dialogs that bring file maps and process modules together into an executable load module. You can define dialogs by using the online dialog compiler.

5. Define the application structure. As an optional step performed at any time before executing the application, you define an application structure that describes the application in terms of functions and responses. In the batch environment, application structures have a special use in creating applications that access input files with multiple record formats. You can define the application structure by using the online application compiler.

6. Execute the application. You execute the batch application by using the runtime system in batch mode. The application can access an CA IDMS/DB database, read from input files, and write to output files. In addition, the runtime system can write input records in error to a suspense file, and can write informational and error messages to a log file. A trace facility is available to help you debug an CA ADS Batch application.
While CA ADS Batch is based on concepts familiar to developers of CA ADS applications, it also includes concepts unique to batch application development. This section discusses the following CA ADS Batch concepts:

- Input and output files and file maps
- Suspense files
- Log files
- Batch dialog structure
- Application structure for applications defined using the application compiler

For more information on the new process commands, see ADS Batch Process Command Language (https://docops.ca.com/display/IDMSCU/ADS+Batch+Process+Command+Language). The runtime system and the CA ADS Batch trace facility are described in Runtime Considerations (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

## Input and Output Files and File Maps

### Input and Output Files

**Input files** are sequential files from which batch transactions are read. **Output files** are sequential files to which batch transactions are written. CA ADS Batch provides access to sequential files, such as disk, tape, card, and printer, and to VSAM entry-sequenced data sets (ESDS) that are accessed sequentially. Records in input and output files (that is, file records) can be any type except variable spanned.

**Note:** CA ADS Batch does not support user and nonstandard tape labels. On input, tape labels are bypassed; on output, they cannot be written.

### Describing Files

You describe files, consisting of file, record, and element entities, in the data dictionary using the IDD DDDL compiler.

**Note:** For more information on describing file, record, and element entities, see the CA IDMS IDD DDDL Reference Section. For more information on specifying file characteristics, see Runtime Considerations (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

### File Maps
A file map is defined for each file record type in an application. File maps are used at runtime to transfer data between file record elements and data fields in variable storage. The diagram below shows two file map definitions.

**Mapin Operation**

An input file map, such as INMAP in the above diagram, transfers data from fields in an input file record to data fields in variable storage. Reading a file record using a map is called a mapin operation.

**Mapout Operation**

An output file map, such as OUTMAP, transfers data from data fields in variable storage to fields in an output file record. Writing a file record using a map is called a mapout operation.

**Components of a File Map**

A file map consists of one external record and zero or more internal records, as follows:

- An external record describes the layout of data in a file. INPUT-RECORD of INPUT-FILE is the external record for map INMAP. OUTPUT-RECORD of OUTPUT-FILE is the external record for map OUTMAP.

- Internal records describe data in variable storage to which external record fields map. WORK-RECORD and DATABASE-RECORD are internal records for both INMAP and OUTMAP.

**Fields Mapping to Themselves**
Fields can map to themselves. For example, in the diagram, IN-FIELD5 maps to itself. On a mapin operation at runtime, IN-FIELD5 in the input buffer is moved into IN-FIELD5 in variable storage. Automatic editing and error handling can be performed on the field; the length of the field, however, cannot be changed.

**Note:** If a field maps to itself, the entire record is available for access and update as an internal record in variable storage; however, only those fields included in the map definition are mapped into or out of variable storage. In the example, INPUT-RECORD is available in variable storage because IN-FIELD5 maps to itself. On a mapin operation, however, data is moved only to IN-FIELD5 of INPUT-RECORD; nothing is moved to the other fields of the record.

**Internal Record Fields**

Map process commands refer to internal record fields. For example, a process module whose dialog is associated with INMAP could issue commands such as the following:

IF FIELD WORK-FIELD1 IS IN ERROR...

MODIFY MAP FOR FIELD DB-FIELD1 EDIT IS ERROR.

A map process command could also reference IN-FIELD5, but could not reference any of the other fields in INPUT-RECORD.

**Input and Output File Maps**

File maps can be used for both input and output. You specify that a file map is for input or output during dialog definition when you associate the map with a dialog. A map can be the input file map for one dialog and the output file map for another dialog. A map can be the input and the output file map for the same dialog. In the diagram, INMAP could be associated with a dialog as an output file map, and vice versa for OUTMAP.

**Accessing Different Data Sets**

The same map can access different data sets. You associate input and output file maps with data sets on a dialog-by-dialog basis. In this way, a single file map can be used to access several data sets in an application.

For example, in an application that copies one data set to another, only one file map is necessary. The map is the input file map for one dialog, and in that capacity, is associated with the input data set. The same map is the output file map for the same or another dialog, and in that capacity, is associated with the output data set.

**Automatic Editing and Error Handling**

Automatic editing and error handling specifications can be part of the map definition. For example, a field can be converted from a DISPLAY field to a COMPUTATIONAL field. Edit and code tables can be used to check fields for valid values and to convert fields from one value to another.

**Response Fields**
A **response field** can be included in the definition of a file map. You specify that a field is a response field by associating it with $RESPONSE during map definition. In the example, IN-FIELD1 is a response field. On a mapin operation at runtime, the value of the response field can determine the next dialog response process or application function that is executed, just as in the online environment.

An output file map can also include a response field. On a mapout operation, the value of the $RESPONSE system-supplied data field is mapped out to external record fields as appropriate.

A response field can be a single field or the concatenation of several fields in a file record. When two or more fields are used to form the response field, you specify the sequence in which the fields are concatenated. For example, a record consisting of fields A through F can have a response field made up of the concatenation of fields D, B, and F, in that order. The maximum length of the concatenated response field is 32 bytes.

Fields that compose a response field must have a usage of DISPLAY. Note that response fields used in applications defined with the application compiler can have a maximum length of eight bytes.

**Excluded Fields**

Fields can be excluded from a map definition. For example, in the diagram, IN-FIELD3 is not included in the definition for INMAP, and OUT-FIELD2 is not included in the definition for OUTMAP. On a mapin operation, data in excluded fields is not mapped into variable storage.

On a mapout operation, the excluded fields are **initialized** according to the following rules:

- Alphanumeric fields not defined with a VALUE clause are initialized to spaces.
- Numeric fields not defined with a VALUE clause are initialized to zero in the proper data format.
- Fields defined with a VALUE clause are initialized to the specified value in the proper data format.

**Note:** Because the output buffer is initialized, you can have an external record with FILLER fields.

For more information on mapping in and mapping out, see **Batch Dialog Structure (see page 18)** and **Application Structure (see page 24)**.

**Suspense Files**

Suspense files store input file records found to contain edit errors at runtime. An input record field can be set in error either at mapin by automatic editing and error handling, or during premap or response process execution by a map modification command.

**Associating a Suspense File with a Dialog**

One suspense file can be associated with each dialog that has an input file. You associate a suspense file with a dialog simply by requesting one, either during dialog definition or at runtime.
Writing to a Suspense File

An input record in error is written to a suspense file at runtime by the dialog that performed the mapin operation on the record. The record can be written either immediately after the mapin operation or on the next mapout operation, as follows:

- **Immediately after the mapin operation**, if the selected response process has an EXECUTE ON EDIT ERRORS specification of NO. The runtime system writes the record to the suspense file, then reads the next input record.

- **On a mapout operation**. The mapout operation is initiated by a WRITE TRANSACTION command. If the dialog's input record is not in error, WRITE TRANSACTION performs a mapout to the dialog's output file. If the input record is in error, WRITE TRANSACTION instead writes the input record to the suspense file.

⚠️ **Note**: You can write the input record to the suspense file even if the record has no fields in error by including the keyword SUSPENSE in the WRITE TRANSACTION command.

⚠️ **Note**: If a suspense record is to be written by a dialog that is not associated with a suspense file, the record is not written. However, error messages are still sent to the log file, which is described later in this section. The suspense file identifier in the messages is ‘********’.

Maximum Number of Error Records

The maximum number of error records that can be written to a suspense file at runtime is a system-generation option (`ADSOBSYS (see page 48)`). By default, an unlimited number of error records can be written. The ADSOBSYS default can be overridden at runtime with control statements for each dialog.

Log Files

Contents

- Log File Information (see page 17)
- Log File Prefix (see page 18)
- Archiving (see page 18)

Log files store error messages and other information produced during application execution. You can specify log file characteristics by using the ADSOBSYS utility; these characteristics can be overridden at runtime using control statements. For more information on specifying log file characteristics, `ADSOBSYS (see page 48)` and Control Statements ([https://docops.ca.com/display/IDMS19/Runtime+Control+Parameters#RuntimeControlParameters-ControlStatements](https://docops.ca.com/display/IDMS19/Runtime+Control+Parameters#RuntimeControlParameters-ControlStatements)).

Print Log Facility
CA ADS Batch provides a print log utility that enables the user to print information from the log based on selection criteria, such as the user ID and the type of record. For more information on this utility, see CA ADS Batch Print Log Utility (see page 59).

The following topics are discussed below:

- Information stored in a log file
- The log file prefix
- Archiving

Log File Information

Types of Log Information

Log files can store the following types of information:

- **User**—Informational messages issued by WRITE TO LOG and CONTINUE commands, and by WRITE TRANSACTION commands that perform a mapout to the dialog's output file. The message is specified in the MESSAGE parameter of the command.

- **Debug**—Debugging information issued by SNAP commands and by the CA ADS Batch trace facility.

- **Edit error**—Error messages issued when an input record is written to the suspense file. The runtime system writes one record consisting of the associated suspense file record number and the first 32 bytes of the input record, followed by one record for each applicable error message. In addition, if the input record was written to the suspense file by a WRITE TRANSACTION command, any message specified in the MESSAGE parameter of the command is also written to the log file as an edit error message.

- **Operator**—Operator informational messages issued by WRITE TO OPERATOR commands. All messages sent to the operator are also sent to the log.

- **Abend**—Abnormal termination messages issued by the ABORT command and by the runtime system when an application abends. Abend messages include snap dumps.

  **Note:** If the abend occurs during a mapping operation, the snap dump goes instead to the z/OS data set associated with the ddname IDMSSNAP file associated with the linkname IDMSSNAP or, in VSE, to SYSLST, as described under JCL (https://docops.ca.com/display/IDMS19/JCL).

- **Statistics**—Statistics records written by the runtime system when dialog statistics are being collected for the application.
Log File Prefix

Information Specified

Each log file record can include a prefix that specifies the following information:

- The date and time the record was written
- The name of the dialog that wrote the record
- The type of record written (for example, error text or statistics)
- The application user id

This information is used by the print log utility to select log records and format them for output.

By default, a prefix is included when the log file output device is tape or disk, and is omitted when the output device is printer. The default can be overridden through the ADSOBSYS utility and/or at runtime with control statements.

For developers creating their own reports from information in the log file, the DSECT for the log file prefix is provided in CA ADS Batch Print Log Utility (see page 59).

Archiving

Unlimited Log Information

Archiving enables z/OS users to archive unlimited log information to tape at runtime. The runtime system writes log records to the first log file. When the file is full, the runtime system writes to the second log file and archives the first log file to tape. When the second log file is full, the runtime system writes to the first log file and the second log file is archived, and so forth.

When Archiving is not Requested

If archiving is not requested, the runtime system performs log file wraparound. Up to two log files can be allocated for an application. If both are full, the runtime system overwrites the log records already written, starting at the beginning of the first log file.

Batch Dialog Structure

Contents

- CA ADS Batch Dialog Components (see page 19)
- Batch Control Events (see page 20)
- Batch Response Field Values (see page 20)
- Batch Dialog Execution (see page 21)
The structure of an CA ADS Batch dialog is similar to that of an CA ADS dialog. Both consist of premap and response processes, maps, work records, and a subschema.

CA ADS Batch Differences

CA ADS Batch dialogs differ in a number of ways. Most importantly, batch dialogs use input and output file maps to transfer data between variable storage and files, while online dialogs use online maps to transfer data between variable storage and online terminals.

Note: Online and batch dialogs are differentiated by the type of map with which they are associated. Dialogs associated with online maps can be executed only in the online environment. Dialogs associated with file maps can be executed only in the batch environment. Mapless dialogs can be executed in either environment.

The following topics are discussed below:

- Components of an CA ADS Batch dialog
- Batch control events
- Batch response field values
- Batch dialog execution

CA ADS Batch Dialog Components

An CA ADS Batch dialog can include the following components:

- Work records
- A subschema
- An input file map
- An output file map
- A premap process
- Response processes

All of these components are optional. However, a dialog must have either a premap process, an input file map, or both. A dialog that has no input file map cannot have any response processes.

Batch Dialog Options
A batch dialog can include most of the dialog options available to online dialogs, such as the request for activity logging. Additional dialog options are available that enable you to specify default z/OS ddnames (z/VSE filenames) for a dialog's input, output, and suspense files. A suspense file is associated with a dialog when you specify its ddname or filename, either during dialog definition or at runtime.

**Batch Control Events**

Batch control events are conditions that arise during file input. Batch control events include:

- **End of file**—The most recent input file read operation resulted in an end-of-file condition.
- **I/O error**—The most recent input file read operation resulted in a physical error condition. Note that errors on output cause the runtime system to abend the application.

**Associating Batch Control Events**

Batch control events can be associated with dialog response processes and application responses. On a mapin operation, the occurrence of a batch control event causes execution of its associated response process or of the application function invoked by its associated application response. For more information on specifying selection criteria for responses and response processes, see Response Definition Screen (see page 33) and "Response Process Definition Screen" in Section5.

**Testing Batch Control Events**

Batch control events can also be tested in premap or response processes, as in the following example:

```
IF $EOF
   LEAVE APPLICATION.
```

For more information on testing for batch control events, see Batch Control Event Status Conditions (https://docops.ca.com/display/IDMSCU/Status+Conditions#StatusConditions-BatchControlEventStatusConditions).

**Batch Response Field Values**

CA ADS Batch extends the online concept of response fields to the batch environment. An input file map can include one or more external fields that make up the map's response field. On a mapin operation, the value of the response field helps to determine the next dialog response process or application function that is executed, just as in the online environment.

For example, an input record can have a response field whose value is ADD, MOD, or DEL. A dialog can be defined with one response process that is executed when the response field value is ADD, another when it is MOD, and another when it is DEL. The response process executed at runtime after the dialog performs a mapin operation depends on the input record's response field value.

**Input Files with Multiple Record Layouts**
Response fields are especially useful in accessing input files that have multiple record layouts, as described under “Application structure” later in this section.

Batch Dialog Execution

Dialog Execution

Batch dialog execution is similar to online dialog execution: a batch dialog executes premap and response processes and performs mapin and mapout operations. The major difference between batch and online dialogs is in the sequence and handling of the mapin (read) and mapout (write) operations.

The diagram below shows a typical batch dialog. The dialog reads from an input file and writes to an output file.

- **Premap Process**
  - The premap process is executed at the beginning of the dialog, unless the dialog’s entry point is its mapin operation. The process executes until it issues a control command, including a READ TRANSACTION or WRITE TRANSACTION command. In the example shown above, a READ TRANSACTION command is issued, which terminates the process and passes control to the mapin operation.

- **Mapin Operation**
  - The mapin operation is performed in any of the following cases:
    - At the beginning of the dialog if the dialog’s entry point is its mapin operation
    - After a READ TRANSACTION command has been issued by a premap or response process
    - After a mapout operation that has been issued by a WRITE TRANSACTION command that does not include the CONTINUE or RETURN keyword
    - After a CONTINUE command has been issued in a dialog without a premap process
  
  The mapin operation performs the following functions:
  1. Reads a record from the dialog’s input file into the file’s input buffer
  2. Maps all correct fields into variable storage according to the dialog’s input file map definition
3. Selects a response process based on batch control event or response field value

**Note:** If the application is defined using the application compiler, the runtime system first examines the response field of the record and passes control to another application function, if required. Application execution using the application compiler is described later in this section.

**Response Process**

The selected **response process** is executed after the mapin operation. The response process executes until it issues a control command, including a batch READ TRANSACTION or WRITE TRANSACTION command. In the example, the response process issues a WRITE TRANSACTION command, which causes a mapout operation.

**Mapout Operation**

A **mapout operation** is performed when a premap or response process issues a WRITE TRANSACTION command. The mapout operation either maps a record to the dialog's output file or writes the input record to the dialog's suspense file, as follows:

- **A record is mapped to the output file** if the dialog's current input file record (if any) contains no errors and if the WRITE TRANSACTION command does not include the keyword SUSPENSE. The mapout operation writes a record to the dialog's output file from variable storage according to the dialog's output file map definition.

- **The input record is written to the suspense file** if the dialog's current input file record contains errors or if the WRITE TRANSACTION command includes the keyword SUSPENSE. Applicable error messages are written to the log file. Note that an input field can be set in error either automatically on mapin or by map modification commands in process code.

**Determining How Control Will Be Transferred**

After the mapout operation, transfer of control is determined by the keyword specified in the WRITE TRANSACTION command that caused the mapout operation:

- **CONTINUE** executes the dialog's premap process.

- **RETURN** returns control to a higher level dialog or application function.

If no keyword is provided, the dialog's mapin operation is performed; this is the case below.

**Note:** Because the WRITE TRANSACTION command passes control to another part of the application, and because the command does not write to both the suspense file and the output file at the same time, it is difficult to write to both files if that is what you want. An alternative is to associate the output file map with a second dialog. The first dialog links to the second, which issues a WRITE TRANSACTION RETURN command. The command writes a record to the output file and returns to the first dialog. The first dialog then issues a WRITE TRANSACTION command that writes the input record to the suspense file.
Control Command Example

Control commands, including READ and WRITE TRANSACTION commands, the READ TRANSACTION and WRITE TRANSACTION commands, affect runtime flow of control. A few of these commands are illustrated below.

IDMSDB--Batch Dialog Execution (2)

Accessing Multiple Input and Output Files

Control commands enable a batch application to consist of multiple dialogs. Through multiple dialogs, an application can access multiple input and output files, as shown below.

IDMSDB--Batch Dialog Execution (3)
Application Execution

Application execution is described below:

1. **DIALOG1's premap process** issues a READ TRANSACTION command, which reads a record from the dialog's input file, then selects a response process.

2. **DIALOG1's response process** issues a LINK command, which passes control to DIALOG2.

3. **DIALOG2's premap process** issues a LINK command, which passes control to DIALOG3.

4. **DIALOG3's mapin operation** (the dialog's entry point) reads a record from the dialog's input file, then selects a response process.

5. **DIALOG3's response process** issues a RETURN command, which returns control to DIALOG2 at the command that immediately follows the LINK command.

6. **DIALOG2's premap process** issues a WRITE TRANSACTION RETURN command, which performs a mapout operation, writing a record to the dialog's output file, and returns control to DIALOG1 at the command that immediately follows the LINK command.

7. **DIALOG1's response process** issues a WRITE TRANSACTION command, which performs a mapout operation, writing a record to the dialog's output file, and performs a mapin operation, reading another record from the dialog's input file and selecting a response process.

For detailed information on the flow of control in a mapin operation, see [Runtime Flow of Control](https://docops.ca.com/display/IDMS19/Runtime+Flow+of+Control).

Application Structure

Contents

- Disallowed Functions (see page 25)
- Application Flow of Control (see page 25)
- Accessing Input Files with Multiple Record Layouts (see page 27)

Defining an Application Structure

An CA ADS Batch application structure can be defined using the application compiler. As with online applications, you define the application structure in terms of functions, responses, task codes, and global records.

Batch application structures differ from online application structures in the following ways:

- Certain functions, such as menu functions, are disallowed.
- Flow of control is slightly different.
- Application structures have a special use in accessing multiple record layouts for input files.
These differences are discussed separately below.

**Disallowed Functions**

The following types of functions are disallowed in a batch application structure:

- Menu functions
- Menu-related system functions, including POP, POPTOP, HELP, FORWARD, and BACKWARD
- Signon system functions, including SIGNON and SIGNOFF
- The ESCAPE system function

**Specifying the Environment**

The application compiler allows you to specify the environment in which an application can be executed, as follows:

- **Batch-only** applications can be executed only in the batch environment. The application compiler prevents you from defining disallowed functions.
- **Online-only** applications can be executed only in the online environment.

**Application Flow of Control**

Flow of control in applications defined using the application compiler is, for the most part, similar for both batch and online applications.

The following special considerations apply to CA ADS Batch applications.

**Selection of Responses**

Application responses are selected on the basis of batch control events and input record response field values. In the online environment, responses are selected on the basis of a control key pressed or a response field value entered by the terminal operator.

As in the online environment, batch application responses invoke application functions; when a response is selected, so is the function it invokes.

**Immediately Executable Functions**

Application functions are, by default, immediately executable. Using the application compiler, you can specify, whether a function is **immediately executable** or **deferred**. The runtime system uses the specification on mapin operations to determine the next dialog response process or application function to be executed. If, on a mapin operation, both a response process and a function are valid selections, transfer of control depends on the specification for the selected function:

- **Immediately executable**—Control passes to the selected function.
Note: An exception is made when the selected function is the same as the current function. In such a case, the response process is executed.

- **Deferred**—Control passes to the selected response process. To pass control to the deferred function, the selected response process can issue an EXECUTE NEXT FUNCTION command.

In the batch environment, all functions are, by default, immediately executable. You can override the default for a function by using the new Response Definition screen of the application compiler. The specification is made for the response that invokes the function.

**Differences Between Batch and Online**

This difference between batch and online flow of control stems from the difference in transaction processing. In the online environment, even if the terminal operator requests transfer to another function, data on the current screen may still require processing by the current dialog before control is passed to the next function. Therefore, the current dialog’s response process takes precedence.

In the batch environment, if the current record’s response field value selects a different function, it is assumed that that function is required to process the current input record. Thus, by default, control passes immediately to that function.

Immediately executable functions enable applications to access input files that have multiple record layouts, as described later in this section.

**Mapin Operations**

Mapin operations are performed by the appropriate functions. For input files that have response fields, the runtime system first examines the response field. If the response field keeps control within the current function, the runtime system maps the record into variable storage and executes a response process. If, instead, the response field selects another function, the runtime system delays the mapin and passes control to the selected function. The next time a mapin operation is performed for that file, the runtime system immediately maps in the record.

**Delayed Mapin**

The major points regarding delayed mapin are as follows:

- Delaying mapin allows control to be passed to the dialog whose map handles the type of record being read.

- The delayed mapin can be performed only if the application is defined using the application compiler.

- The application functions should be defined as immediately executable, as they are by default.

- A dialog receiving control after a delayed mapin must perform a mapin operation to map the record into variable storage.

Delayed mapin enables applications to access input files that have multiple record layouts, as described below.
Accessing Input Files with Multiple Record Layouts

Application structures have a special use in CA ADS Batch in enabling applications to access input files with multiple record layouts.

For example, consider an application that reads a file containing two types of records stored together: TYPE1 records and TYPE2 records. Each type of record has its own record layout. Additionally, the first two bytes of each record identify the record type: T1 for TYPE1 records and T2 for TYPE2 records.

You define a map for each record layout and associate each map with a dialog (DIALOG1 and DIALOG2). At runtime, the runtime system must use the proper dialog for each input record. Since the sequence of record layouts on input is not predictable, the runtime system must know ahead of time which dialog to execute for each possible record layout. This is done by defining an application structure using the application compiler.

The diagram below shows the application structure and provides sample input data. Response T1 invokes function FUNCTION1, which executes dialog DIALOG1. Response T2 invokes function FUNCTION2, which executes dialog DIALOG2. Both responses are valid from both functions.
Application Execution

Application execution is described below:

1. FUNCTION1 executes DIALOG1. DIALOG1 performs a mapin operation.

2. The runtime system examines the response field of the first input record; it is a T1 record. Since T1 invokes the current function, the runtime system maps the record into variable storage and selects and executes a response process. The response process processes the record, then issues a READ TRANSACTION command, which terminates the current process and performs another mapin operation.
3. The runtime system examines the response field of the second input record; it, too, is a T1 record. The record is mapped in and the response process is executed. The response process issues a READ TRANSACTION command, which terminates the current process and performs another mapin operation.

4. The runtime system examines the third input record; it is a T2 record. Since T2 invokes FUNCTION2, control passes immediately to FUNCTION2, which executes DIALOG2. The record has not yet been mapped into variable storage. DIALOG2 performs a mapin operation.

5. The runtime system immediately maps the third record into variable storage (its response field was already examined) and selects and executes a response process. The response process issues a WRITE TRANSACTION command, which terminates the current process, maps a record to the dialog's output file, then performs a mapin operation.

6. The runtime system examines the response field of the fourth input record; it is a T1 record. The runtime system immediately invokes FUNCTION1, which executes DIALOG1. DIALOG1 performs a mapin operation.

7. The runtime system immediately maps the fourth record into variable storage, then selects and executes a response process. The response process issues a READ TRANSACTION command, which terminates the current process and performs a mapin operation.

8. The runtime system attempts to examine the response field of a fifth input record, but encounters an end-of-file condition. Two ways for the application to handle an end-of-file condition are described below:
   - The application can be defined so that the end-of-file condition selects a function whose dialog handles the condition.
   - A response process in DIALOG1 and DIALOG2 can be associated with the end-of-file condition; the response process would be selected and executed when the condition occurred.

Application Compiler

The application compiler enables you to predefine an application structure in terms of functions, responses, task codes, and global records. This section presents all of the application compiler screens, accompanied by notes on fields and functionality specific to CA ADS Batch.

⚠️ **Note:** For more information about accessing these screens and descriptions of the fields, see the *CA ADS Reference Section.*
General Options Screens

**Application name:** METAPPL1  **Version:** 1

**Description** . . . MIKE T’S APPLICATION

**Maximum responses** . . . . . . . . . . . . . . 500

**Date format** . . . . . . . . . . . . . . . . . . . 1 1. mm/dd/yy 2. dd/mm/yy 3. yy/mm/dd 4. yy/ddd

**Execution environment** . . . . . . . . . . . . 2 1. Online 2. Batch

**Default execution mode** . . . . . . . . . . . . 1 1. Step 2. Fast

**Default print destination** . . . . . . . . . .

**Default print class** . . . . . . . . . . . . . . 1

Enter F1=Help F3=Exit F4=Prev F5=Next F8=Fwd

**Application name:** METAPPL1  **Version:** 1

**Security class** . . . . . . . . . . . . . . . . . . 42

**Menus are** . . . . . . . . . . . . . . . . . . 1 1. Not used 2. Security tailored 3. Untailored

**Signon is** . . . . . . . . . . . . . . . . . . 1 1. Not used 2. Optional 3. Required

**Signon function is** . . . . . .

Enter F1=Help F3=Exit F4=Prev F5=Next F7=Bkwd

**Batch Considerations**

On the second General Options screen you can specify the environment in which the application can be executed, as follows:

- **BATCH** specifies that the application can be executed only in the batch environment.

  If BATCH is specified, the application compiler prevents you from entering online-only specifications, including:

  - Security specifications
Menu- and signon-related system functions in the Function invoked field on the Response /Function List screen. These functions include POP, POPTOP, HELP, FORWARD, BACKWARD, SIGNON, and SIGNOFF. Additionally, the ESCAPE system function cannot be specified.

Menu and menu/dialog functions on the Response/Function List screen and the Function Definition screen.

ONLINE specifies that the application can be executed only in the online environment. The application compiler accepts all valid specifications on all screens.

Global Records Screen

The Global Records screen, shown below, enables you to specify records that are to be made available to all functions in an application.

Sample Screen

<table>
<thead>
<tr>
<th>Record name</th>
<th>Version</th>
<th>Drop record (/)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ADSO-APPLICATION-GLOBAL-RECORD</td>
<td>1</td>
<td>_</td>
</tr>
<tr>
<td>2. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
<tr>
<td>3. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
<tr>
<td>4. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
<tr>
<td>5. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
<tr>
<td>6. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
<tr>
<td>7. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
<tr>
<td>8. __________________________</td>
<td>______</td>
<td>_</td>
</tr>
</tbody>
</table>

Enter F1=Help F3=Exit F4=Prev F5=Next F7=Bkwd F8=Fwd

Note: For more information on the Global Records screen, see the CA ADS Reference Section.

Task Codes Screen

The Task Codes screen, shown below, enables you to specify task codes that initiate execution of the application.

Sample Screen
Response/Function List Screen

Sample Screen

Application name: METAPPL1 Version: 1

Select (/) Response name Assigned key Select (/) Function name/type(1,2,3)* Program/Dialog name
_ R1 ENTER _ F1 / 1 JPKSQLD1
_ R3 PF02 _ FUNC3 / 1 STEVEDLG
_ R2 PF02 _ FUNC2 / 3 ________
_ R4 PF04 _ FUNC4 / 1 DIAL4
_ R5 PF05 _ FUNC5 / 1 DIAL5
_ LINKOLQR PF06 _ LINKOLQF / 2 IDMSOLQS

* Type: 1. Dialog 2. Program 3. Menu

Enter F1=Help F3=Exit F4=Prev F5=Next F6=Search F7=Bkwd F8=Fwd

Batch Considerations

Special considerations apply to the following fields:
- **Response name**—Enables you to specify an input record response field value that initiates an application function.

  **Note:** If the response field for an input record is the concatenation of several fields, the response name you specify on the Response Definition screen must include any embedded blanks that would occur in a concatenation. For example, if a response field is the concatenation of two fields, the first being six bytes long, you would type the following response name to specify a response field value of ADD for the first field and E for the second field:

  ADD   E

- **Assigned key**—This field enables you to specify a batch control event. Valid batch control events are as follows:

  - **EOF**—The most recent input file read operation resulted in an end-of-file condition.
  - **IOERR**—The most recent input file read operation resulted in a physical input-error condition. Note that in CA ADS Batch, an output error causes the runtime system to terminate the application.

- **Function name**—The POP, POPTOP, HELP, FORWARD, BACKWARD, SIGNON, SIGNOFF, and ESCAPE system functions are disallowed entries for batch-only applications. In batch-only applications, you cannot specify a function type of 3 (Menu).

  **Note:** For more information on the Response/Function List screen, see the *CA ADS Reference Section*.

**Response Definition Screen**

The Response Definition screen, shown below, enables you to define application responses.

**Sample Screen**

```
Response Definition

Application name: TEST1   Version: 1
Response name: QUIT       Drop response (/)
Function invoked: QUIT     Description . . . ______________________________
Response type . . . . . . 2 1. Global  2. Local
Response execution . . . 2 1. Immediate 2. Deferred
Assigned key . . . . . . . PF01
Control command . . . . 1 1. Transfer 2. Invoke
                           3. Link 4. Return
                           5. Return continue 6. Return clear
                           7. Return continue clear 8. Transfer nofinish
```
CA IDMS - 19.0

9. Invoke nosave 10. Link nosave

Enter F1=Help F3=Exit F4=Prev F5=Next

**Batch Considerations**

- **Response execution** specifies whether the invoked function is immediately executable or deferred. In the batch environment, all functions are, by default, immediately executable. **Response execution** allows you to override these defaults.
  - You override the default by entering 1 (immediately executable) or 2 (deferred) in the data field that immediately follows **Response execution**.
  - For a discussion of immediately executable and deferred functions, see "Application Structure" in CA ADS Batch Concepts (see page 11).

- **Control command** displays the control command associated with the current application response.

⚠️ **Note:** For more information on control commands, see the *CA ADS Reference Section*.

You can select one of the following specifications by entering the appropriate number in the data field immediately following the prompt.

---

**Function Definition Screen**

The Function Definition screen, shown below, enables you to provide basic information on a function.

**Sample Screen**

```
Function Definition (Dialog)

Application name: TEST1 Version: 1
Function name: F2 Drop function (/) _
Description . . . UNDEFINED
Associated dialog . . . D2 User exit dialog . . . ________
Default response . . . ________
Valid response(/) Response Key Function Valid response(/) Response Key Function
_ ADD PF02 F2 _ __________ __________
_ QUIT PF01 QUIT _ __________ __________
_ __________ __________ _ __________ __________
_ __________ __________ _ __________ __________
_ __________ __________ _ __________ __________
```

Enter F1=Help F3=Exit F4=Prev F5=Next F7=Bkwd F8=Fwd
Dialog Compiler

The dialog compiler is used to define dialogs that perform transaction input and output, database retrieval and update, and any required processing within an application.

**Associated Map**

The environment within which a dialog can be executed depends on the map with which it is associated, as follows:

- A dialog associated with an **online map** can be executed only in the online environment.
- A dialog associated with a **file map** can be executed only in the batch environment.
- A **mapless dialog** can be executed in either environment.

At runtime, if the runtime system encounters a disallowed dialog, it abends the application.

**Process Modules**

Process modules associated with batch dialogs can also include online-only commands and command parameters, and vice versa. The dialog compiler, when compiling a process module, accepts both types of commands, regardless of the environment of the dialog. This allows a process module to be used in both the online and batch environments. If, however, the runtime system encounters a disallowed command or command parameter, the application abends. For more information on process commands in the batch and online environments, see *CA ADS Batch Concepts* (see page 11).

A dialog can be defined using the dialog compiler in either online or batch mode.

**Note:** Do not confuse batch and online definition modes with batch and online execution modes. The terms "batch" dialog and "online" dialog refer to the environment in which the dialog can be executed. Batch dialogs and online dialogs can be defined using the dialog compiler in online or batch mode, whichever is most convenient.

**Batch Considerations**

- You can associate a dialog with one input file map and one output file map.
- You can specify:
  - The z/OS ddnames (z/VSE filenames) of the dialog's input, output, and suspense files
You can specify:

- That the response process currently being defined is the dialog's default response process
- That batch input record response field values and batch control events to associated with the response process

**Dialog Compiler in Online Mode**

**Contents**

- Main Menu Screen (see page 36)
- Map Specifications Screen (see page 37)
- Database Specifications (see page 38)
- Options and Directives Screen (see page 38)
- Records and Tables Screen (see page 39)
- Process Modules Screen (see page 40)

The dialog compiler in online mode displays screens on which you specify information on the dialog you are defining. All screens are presented on the following pages, accompanied by notes on batch definition functionality.

**Note:** For more information on accessing screens and a description of each screen, see the CA ADS Reference Section.

**Main Menu Screen**

The Main Menu screen, shown below, enables you to specify basic information on a dialog.

**Sample Screen**

```
._____________________________________________________________________________.
CA-ADS Online Dialog Compiler
CA

Dialog name . . . . . . . . ____________
Dialog version . . . . . . ______
Dictionary name . . . . . ______
Dictionary node . . . . . ______
Screen . . . . . . . . . . . . . 1. General options
                           2. Assign maps
                           3. Assign database
                           4. Assign records and tables
                           5. Assign process modules
```

13-Jan-2018 36/114
Map Specifications Screen

The Map Specifications screen, shown below, enables you to specify information on the map associated with a dialog.

Sample Screen

Map Specifications

Dialog JPKD10 Version 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map name</td>
<td>____________</td>
</tr>
<tr>
<td>Input map</td>
<td>____________</td>
</tr>
<tr>
<td>Version</td>
<td>___</td>
</tr>
<tr>
<td>Label</td>
<td>____________</td>
</tr>
<tr>
<td>Paging options</td>
<td>_1. Wait</td>
</tr>
<tr>
<td></td>
<td>2. No Wait</td>
</tr>
<tr>
<td></td>
<td>3. Return</td>
</tr>
<tr>
<td>Paging mode</td>
<td>_Update</td>
</tr>
<tr>
<td></td>
<td>_Backpage</td>
</tr>
<tr>
<td>Suspension file label</td>
<td>____________</td>
</tr>
</tbody>
</table>

Enter F1=Help F3=Exit F4=Prev F5=Next F6=Switch Protection

Use the F6 key to turn protection for the left and right side of the screen on and off. For example, press F6 if you want to enter information on the right side of the screen; press it again to protect the information that is displayed.

Batch Considerations

The Map name field on the screen allows you to associate map with the dialog and, by doing this, allows you to specify the environment in which the dialog can be executed, as follows:

- **Input map** specifies the name of an input file map and indicates that the dialog can be executed only in the batch environment.
- **Output map** specifies the name of an output file map and indicates that the dialog can be executed only in the batch environment.
- **Version** specifies the 1- to 4-digit version number of the corresponding map. If no version number is specified, Version defaults to 1.
- **Label** allows you to specify the z/OS ddname (z/VSE filename) of a batch dialog's input file map or output file map. Specifications made in these fields can be overridden at runtime.
• **Suspense file label** allows you to specify the z/OS ddname (z/VSE filename) of a batch dialog's suspense file. Specifications made in this field can be overridden at runtime. Runtime labels for an input map and suspense file can be specified only if the dialog is associated with an input file map. A runtime label for an output map can be specified only if the dialog is associated with an output file map.

**Note:** When you supply a runtime label for a suspense file, either during dialog definition or at runtime, you implicitly specify that a suspense file is required for the dialog.

A dialog associated with an online map cannot be associated with an input or output file map, and vice versa. A dialog can be associated with both an input and an output file map. A dialog not associated with a map is called a mapless dialog and can be executed in both the batch and online environments.

**Database Specifications**

The Database Specifications map is shown below.

**Sample Screen**

```
Database Specifications
Dialog  NAME1  Version  1

Subschema
Schema
Version
Access Module
SQL Compliance
1. ANSI-standard SQL
2. FIPS
Date Default Format
Time Default Format
1. ISO
2. USA
3. EUR
4. JIS
```

Enter F1=Help  F3=Exit  F4=Prev  F5=Next

**Options and Directives Screen**

The Options and Directives screen, shown below, enables you to specify special options for a dialog.

**Sample Screen**

```
Options and Directives
Dialog  JPKTD10  Version  1
```
Batch Considerations

Regardless of the entry point specification, a dialog without a batch input file map begins with its premap process. A dialog without a premap process begins with its first mapping operation.

Records and Tables Screen

The Records and Tables screen, shown below, enables you to associate work records with the dialog and to assign the new copy attribute to records known to the dialog.

Sample Screen

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Work</th>
<th>New copy</th>
<th>Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKB-INTERFACE-RECORD</td>
<td>1</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>OTHER-REC</td>
<td>1</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>ADSC-SCRATCH</td>
<td>1</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>ADSC-PROD</td>
<td>1</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>ADSC-DB</td>
<td>2</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>ADSC-TR</td>
<td>1</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
<tr>
<td>ADSC-PROTO-SYNTAX</td>
<td>1</td>
<td>/</td>
<td>_</td>
<td>_</td>
</tr>
</tbody>
</table>

Note: For more information on the Records and Tables screen, see the CA ADS Reference Section.
Process Modules Screen

The Process Modules screen, shown below, enables you to associate a process (premap, response, or declaration) with the dialog.

**Sample Screen**

<table>
<thead>
<tr>
<th>Process Modules</th>
<th>Page 1 of 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog NAME1</td>
<td>Version 1</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
</tbody>
</table>

* Type: 1=Declaration  2=Premap  3=Response  4=Default Response

DC498166 Neither a map nor premap are defined

Enter F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd

⚠️ **Note:** For batch dialogs, if the Value for an input record is the concatenation of several fields, the response value you specify on the Process Modules screen must include any embedded blanks that would occur in a concatenation. For example, if a response value is the concatenation of two fields, with the first being six bytes long, you would type the following value to specify a response field value of ADD for the first field and E for the second field:

ADD   E

**Batch Considerations**

**Key** specifies the batch control event that initiates the response process at runtime. Valid batch control events are listed below:

- **EOF**—The most recent input file read operation resulted in an end-of-file condition.

- **IOERR**—The most recent input file read operation resulted in a physical input-error condition. Note that in CA ADS Batch, output errors cause the runtime system to terminate the application.
Dialog Compiler in Batch Mode (ADSOBCOM)

ADSOBCOM enables you to compile a dialog in batch mode.

Note: For more information on complete syntax and syntax rules for ADSOBCOM, see the CA ADS Reference Section.

CA IDMS/DC Mapping Facility

The CA IDMS/DC mapping facility enables you to define online maps and file maps. Maps can be compiled in both online mode and batch mode. A file map is not in itself an input or output file map. Its use within a dialog, as specified during dialog definition, determines whether it is an input file map, an output file map, or both.

The online and batch modes of the mapping facility are described separately below, following discussions of batch automatic editing and error handling and variable array processing.

Batch Automatic Editing and Error Handling

The automatic editing and error handling facility operates in the batch environment in much the same manner as it does in the online environment. On a mapin operation, the facility edits and performs error checking on data being transferred from an input file to variable storage. On a mapout operation, the facility edits data being transferred from variable storage to an output file.

The automatic editing and error handling facility in batch mode differs in the following ways.

Error Handling

In the online environment, the terminal operator’s screen is redisplayed with error messages. In the batch environment, the error record is written to a suspense file and error messages are written to the log file.

Automatic Editing and Error Handling Options

In the batch environment, several automatic editing and error handling options are not applicable. Options you can specify for file map fields include:

- An external picture
An edit table

A code table

The data transfer option on input

The data transfer option on output

A user-written edit module on input

A user-written edit module on output

An error message text or code

You can specify these options in MAPB on the User-defined Edit Modules screen, the Additional Edit Criteria screen, and the Map Read/Write options screen. With the batch mapping compiler, you use the MFLD statement.

Implied Decimal Point and Sign

The external picture of a file map field can include an implied decimal point (represented by the symbol V) and an implied sign (represented by the symbol S). This differs from map fields in the online environment, in which external pictures, which represent how fields are to be displayed on a screen, cannot include implied decimal points and signs.

Data Types

In the batch environment, the data type of an external field can be display, zoned decimal, packed decimal (COMPUTATIONAL-3) or binary (COMPUTATIONAL). In the online environment, the data type must be display.

Null Values

In the online environment, if the terminal operator enters nothing in a field, nothing is mapped into the associated field in variable storage. In the batch environment, unless the input data transfer option is set to NO, all input record fields included in the map definition are mapped into variable storage.

Variable Array Processing

Variable array records are records that contain a field that occurs a certain number of times depending on the value of another field in the record. An example is shown below:

SAMPLE-RECORD
03 FIELDA PIC s9(4) USAGE IS COMPUTATIONAL.
03 FIELDB PIC X(20) OCCURS 3 TO 10 TIMES
   DEPENDING ON FIELDA.

Processing

When the external record of a file map includes a variable array, CA ADS Batch performs the following processing:
On a mapin operation, the runtime system maps all fields included in the map definition into variable storage, even array fields that do not actually exist for the current input record. For example, in SAMPLE-RECORD, if FIELDB occurs seven times, the eighth, ninth, and tenth occurrences of FIELDB do not exist; however, the runtime system maps in whatever is in the record buffer for these occurrences. The record's control field (that is, FIELDA in the example), should be included in the map definition; otherwise, the field will not be mapped in and the application will not be able to distinguish the real data from the meaningless data in the array.

On a mapout operation, the runtime system always maps out all fields that are included in the map definition, regardless of the value of the control field. The maximum record length for the map's associated external record is always written.

Note: If a field contains meaningless data that cannot be mapped out successfully (that is, fails the automatic editing stage), the application aborts.

Online Mapping Facility

Contents
- Map Definition Screen (see page 44)
- Extended Map Definition Screen (see page 44)
- File Field Selection Screen (see page 45)
- File Field Edit Screen (see page 46)

The online mapping facility (MAPB) enables you to compile maps in online mode. You use the following screens to define a map:

1. **Main Definition screen**—Used to specify basic information on the map, including the map name and the dictionary in which it resides.

   Note: The Main Definition screen records are called **internal records** and refer to records in variable storage. A map's **external record** is the record that describes a layout of data in the input or output file. Record elements in an external record map to data fields in internal records. If an external record element maps to itself, you do not specify the external record on the Map Definition screen.

2. **Field Definition screen**—Used to specify the external record. If the external record participates in several files, you also specify the applicable file.

   Note: Once an external record has been associated with the map, you must explicitly select it to display it.
3. **File Field Selection screen**—Used to select record elements in the external record for participation in the map and for editing.

4. **File Field Edit screen**—Used to associate an external record field with an internal record field and to specify special editing and error handling characteristics.

5. **Map Definition screen**—Used to compile the file map load module.

The MAPB screens are shown below, accompanied by notes on new and changed fields and functionality.

### Map Definition Screen

The Map Definition screen, shown below, enables you to provide basic information on the map.

**Sample Screen**

```
CA IDMS/DC FILE MAPPING REL nn.n *** MAP DEFINITION *** volser

ACTION: MAPNAME: JMAOUT VER: 1 DICTNAME: DICTNODE:
EDIT(Y/N): Y USING RECORDS

RECORD NAME VER ROLE NAME DEL

COPY FROM MAPNAME: VER: COPY ACTION (ALL/FMT):

SELECT NEXT FUNCTION:
- EXTENDED MAP DEFINITION
- FIELD SELECTION
- ADDITIONAL RECORDS
- FILE FIELD SELECTION
- FILE FIELD EDIT
- QUIT
- SUSPEND

Batch Considerations

The Copy Format option on the Add activity on the action bar on the Map Definition screen has no meaning for file maps. If you specify Format for file maps, MAPB changes the action to All.

### Extended Map Definition Screen

The Extended Map Definition screen, shown below, enables you to provide further basic information on the map, including the external file and record to be associated with a file map.
Batch Considerations

Batch fields on the screen are described below:

- **Record name** specifies the name of an external record to be associated with a file map. The record must be described in the data dictionary and included within a file.

- **File name** specifies the name of the external file in which the named record participates. If no file name is provided, MAPB uses the first file in the dictionary's file-record set for the named record. Note that the record's file description in the data dictionary need not include file characteristics; these can be specified instead at runtime. Nevertheless, MAPB displays a warning message, such as the following:

  FILE TYPE NOT DEFINED TO IDD
  DETERMINED AT EXECUTION

- **Version** specifies the 1- to 4-digit version number of the corresponding record or file name. If Version is not specified, it defaults to the data dictionary default version number, as defined by the DDDL SET OPTIONS statement.

File Field Selection Screen

The File Field Selection screen, shown below, enables you to select external record elements for inclusion in the map definition and for editing on the File Field Edit screen. Fields not included in the map definition are not mapped in or mapped out at runtime.

The File Field Selection screen lists all of the fields in the external record. If the record contains more fields than fit on the screen, MAPB builds additional pages to accommodate all the fields. You can display other pages of the screen by specifying a page or by pressing ENTER to display the next page.

You select fields on the File Field Selection screen by keying a nonblank, non-underscore character (except the D character) at the underscore below the SELECT prompt that corresponds to the desired field. You cannot select both a group field and any of its subordinate-level fields.

By selecting an external field for inclusion in the map, you also select it for editing on the File Field Edit screen, displayed later in the map definition. Once a field has been selected and edited, you can again use the File Field Selection screen to reselect the field for editing or to deselect the field from the map definition.
Batch Considerations

Specific fields on the screen are described below:

- **SELECT** specifies that the corresponding record field is to be included in the map definition. To select a field, key a nonblank, non-underscore character (except the D character) at the underscore below the SELECT prompt that corresponds to the desired field. You cannot select both a group field and any of its subordinate-level fields.

  - Selecting a field also selects it for editing on the File Field Edit screen. To reselect a field after it has been edited, key another nonblank, non-underscore character (except for D) over the X.
  - To deselect a selected field, key a D, underscore, or blank over the X.

File Field Edit Screen

The File Field Edit screen, shown below, enables you to associate a selected external record field with an internal record field and to specify special editing and error handling characteristics. The File Field Edit screen is displayed once for every external field selected for editing on the File Field Selection screen.

> **Note:** An error message that you specify for a map field is written to the log file at runtime if the field is in error and the input record is being written to a suspense file.
INTERNAL FIELD
DFLD : LAST-NAME
OF REC : JMAOUT

EDIT TABLE: VER: _ (LINK Y/N) (VALID Y/N/D-DICT)
CODE TABLE: VER: _ (LINK Y/N) Y (EDIT Y/N)

FOR INPUT Y (DATA Y/N)
<EDIT MODULE:_ (WITH EDIT B/A/N)>
FOR OUTPUT Y (DATA Y/N/E)
<EDIT MODULE:_ (WITH EDIT B/A/N)>
N (ERROR MSG T-TEXT/I-ID/N-NULL)  PREFIX: ID: TEXT:

Batch Considerations

Specific fields on the screen are described below:

- **EXTERNAL FIELD** identifies the external record field being edited and specifies its external picture, as described below:
  
  - **FIELD** specifies the field name. This field is protected.
  
  - **OCCURS** specifies the occurrence of the field if the field is multiply-occurring. This field is protected. Levels of occurrences are separated by commas.
  
  - **OF REC** specifies the external record to which the field belongs. This field is protected.
  
  - **VER** specifies the version number of the external record. This field is protected.
  
  - **FILE-PIC** specifies the external picture of the file field. You can modify the picture for use by automatic editing. If the external field is to map to itself, you cannot specify a picture that changes the field’s length.
    
    **Note:** For more information on specifying external pictures, see the *CA IDMS Mapping Facility Section*.

- **INTERNAL FIELD** identifies the internal record field with which the external record field is to be associated, as described below:
  
  - **DFLD** specifies the name of an internal record field.

  **Note:** If no internal field is specified, the external field maps to itself, from the record buffer into variable storage, as described under "Input and Output Files and File Maps" in *CA ADS Batch Concepts* (see page 11).

Alternatively, you can enter $RESPONSE (or $R) to specify that the associated external field is a response field. Several external fields can be designated as response fields. In this case, the response field is composed of the concatenation of all the fields designated as response fields. The maximum length of a concatenated response field is 32 bytes. For more information on response fields, see "Batch Dialog Structure" in *CA ADS Batch Concepts* (see page 11).
- **OCCURS** specifies the occurrence of the field, if any. Up to three levels can be specified. Levels must be separated by commas.
  For a response field composed of the concatenation of several external fields, OCCURS is used to specify the sequence of the external field in the concatenation. By default, external fields are concatenated in the order in which they were associated with $RESPONSE.

- **OF REC** specifies the record to which the external field belongs. OF REC is required only if DFLD is not unique among the fields in the records associated with the map.

- **VER** specifies the version number of the record. This field is required only if more than one version of the internal record is associated with the map.

**Batch Map Compiler (RHDCMAP1)**

RHDCMAP1 enables you to compile maps in batch mode. To define a file map in batch mode, you use automatic panel definition statements (MAP AUTOPANEL and MFLD).

⚠️ **Note:** You cannot define a file map using the section panel definition statements.

Syntax for the MAP AUTOPANEL and MFLD statements can be found in the *CA IDMS Mapping Facility Section*.

**ADSOBSYS**

ADSOBSYS is a utility that builds a load module (ADSOOPTI) that supplies system generation parameters to ADSOBCOM and the CA ADS Batch runtime system. ADSOBSYS must be run once for each DC/UCF system at an installation. Additionally, ADSOBSYS must be run whenever Application Development System system generation parameters are changed.

**Control statements**

Control statements are available that enable you to specify default parameters for CA ADS Batch application execution. At runtime, you can override any of these parameters by using control statements, as described in *Runtime Considerations* (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

The control statements must follow the SYSTEM statement in the ADSOBSYS job stream. The ADSOBSYS parameters shown in this appendix affect only the CA ADS Batch runtime environment. They do not affect the CA ADS online runtime environment. For example, the STATISTICS parameters specifies how statistics are to be collected only for an CA ADS Batch application. You specify the CA ADS online environment at system generation in the ADSO statement.

Syntax for the CA ADS Batch control statements is shown below:
• PRIMARY POOL IS primary-pool-size
  Specifies the size, in bytes, of the primary buffer pool. Primary-pool-size is an integer in the range 0 through 2,147,483,647. The default is determined at system generation; the system generation default is 4,000.

  **Note:** For more information on specifying primary pools, see the discussion of the ADSO statement in *CA IDMS System Generation Section*.

• SECONDARY POOL IS secondary-pool-size
  Specifies the size, in bytes, of the secondary buffer pool. The secondary pool is allocated from the system storage pool when the primary buffer pool becomes full. Secondary-pool-size is an integer in the range 0 through 2,147,483,647. The default is determined at system generation; the system generation default is 2,000.

• DIALOG STATISTICS OFF/ON
  Specifies whether dialog statistics are to be collected during application execution.
  OFF specifies that statistics are not to be collected.
  ON specifies that statistics are to be collected.
  The default is determined at system generation; the system generation default is OFF.
  Parameters included in the ON clause are as follows.

  ▪ ALL/SELECTED
    Specifies whether dialog statistics are maintained for all dialogs or for only selected dialogs in an application. The default is determined at system generation.
    Dialogs are selected at runtime through the DIALOG STATISTICS control statement.

  ▪ CHECKPOINT INTERVAL IS checkpoint-interval
    Specifies the frequency with which dialog statistics are to be written to the log file. Statistics are written to the log file once every time statistics are accumulated for the checkpoint-interval time. CA ADS Batch accumulates statistics every time a dialog issues a control command. The default checkpoint interval is determined at system generation; the system generation default is 200.

• ROUTING CODES ARE (route-code)
  (z/OS) specifies the OS operator-message routing codes, as described in the applicable operating system supervisor services and macro instructions. This parameter supplies the value of the ROUTCDE parameter for WTO (write-to-operator) macros issued by the system. Route-code is an integer in the range 1 through 16; the default is 11. Multiple routing codes must be separated by commas.

• DESCRIPTOR CODES ARE (descriptor-code)
  (z/OS) specifies the z/OS operator-message description code, as described in the applicable operating system supervisor services and macro instructions. This parameter supplies the descriptor code to the DESC parameter for WTO (write-to-operator) macros issued by the system. Descriptor-code is an integer in the range 1 through 16; the default is 7.
• **DIALOG HAS MAXERR COUNT OF** *maximum-errors*
  Specifies the maximum number of error records that can be sent to the suspense file of any dialog. If this number is exceeded at runtime, the runtime system terminates the application. For example, if the maximum error count is 1, the runtime system terminates the application when the second error record is to be written. If the maximum error count is 0, the runtime system allows an unlimited number of error records to be sent to a suspense file. *Maximum-errors* is an integer in the range 0 through 32,767; the default is 0.

• **OPERATOR SHUTDOWN IS ENABLED/DISABLED**
  Specifies whether the operator can send a request to the runtime system to terminate an application. If operator shutdown is enabled, the runtime system begins application execution by issuing a WTOR (write-to-operator with reply) macro. At any time during application execution, the operator can issue a SHUTDOWN command, as described in [Runtime Considerations](https://docops.ca.com/display/IDMS19/Runtime+Considerations).

  The SHUTDOWN command causes the runtime system to terminate the application with an optional dump. If shutdown is disabled, the operator can terminate the application only by abnormally terminating the runtime system. The default specification is DISABLED.

• **MAXIMUM LINKS IS** *maximum-links*
  Specifies the maximum number of dialog levels that can be established by each CA ADS Batch application thread. *Maximum-links* is an integer in the range 0 through 32,767. The default is determined at system generation; the system generation default is 10.

• **COMMIT WHEN FILES OPEN CAUSES NOACTION/WARNING/ABEND**
  Specifies the action to be taken when a database commit is to be performed before all files used in the application have been closed, as follows:

  - **NOACTION** specifies that no action is taken.
  - **WARNING** specifies that a warning message is sent to the log file.
  - **ABEND** (default) specifies that the application is abended.

• **LOG**
  Specify defaults for the log file. All LOG statement parameters are optional. The parameters are as follows.

  - **FILE1 IS** *adsloga/run-time-label*
    Specifies the runtime label (z/OS ddname, z/VSE filename) of the primary log file.

  - **COUNT1 IS** *file1-maximum-records*
    Specifies the number of log records that are written to the primary log file before the file is considered full.

      *file1-maximum-records* is reached at runtime, the runtime system switches to the secondary log file or, if no secondary log file is allocated, wraps around to the beginning of the primary log file. If archiving is requested, the runtime system archives the primary log file to tape. If 0 is specified, no predefined limit is placed on the number of records written to the primary log file. If space for a disk log file is exceeded at runtime, the runtime system abnormally terminates the application.

      *File1-maximum-records* is an integer in the range 1 through 99999; the default is 0.

  - **FILE2 IS** *run-time label*
    Specifies the runtime label of the secondary log file.
- **COUNT2 IS file2-maximum-records**
  Specifies the number of log records that are written to the secondary log file before the file is considered full.
  If file2-maximum-records is reached at runtime, the runtime system switches back to the primary log file. If archiving is requested, the runtime system archives the secondary log file to tape.
  If 0 is specified, no predefined limit is placed on the number of records written to the secondary log file. If space for a disk log file is exceeded at runtime, the runtime system abnormally terminates the application.
  File2-maximum-records is an integer in the range 1 through 99999; the default is 0.

- **ARCHIVE/NOARCHIVE**
  (z/OS only) specifies whether log file archiving is to be performed at runtime when a log file is full. For more information on log file archiving, see Log Files (see page 16). The default is NOARCHIVE.

- **COMPRESS/NOCOMPRESS**
  Specifies whether log records are compressed in the log file to save space at runtime. If neither COMPRESS nor NOCOMPRESS is specified, the runtime system uses the z/OS and z/VSE device-type assignment to determine whether to compress the records:
  - In z/OS, records are not compressed if the assignment is SYSOUT; otherwise, records are compressed.
  - In z/VSE, records are not compressed if the device type is PRINTER (as specified using ADSOBSYS or at runtime); otherwise, records are compressed.

- **PREFIX/NOPREFIX**
  Specifies whether a prefix is to precede each log record. Prefixes are required by the print log utility. If neither PREFIX nor NOPREFIX is specified, the runtime system uses the z/OS and z/VSE device-type assignment to determine whether to include a prefix:
  - In z/OS, a prefix is not included if the assignment is SYSOUT; otherwise, a prefix is included.
  - In z/VSE, a prefix is not included if the device type is PRINTER (as specified using ADSOBSYS or at runtime); otherwise, a prefix is included.

- **SUBSCHEMA-LR-CTRL SIZE IS lrc-block-size**
  Specifies the space, in bytes, reserved for the logical record request WHERE clause (PXE), which is passed internally in the LRC block. The default is 512.
  The larger the WHERE clause, the more space is required for the PXE. The default of 512 is large enough to include approximately 32 operators, operands, and literals.
  The SUBSCHEMA-LR-CTRL SIZE parameter enables you to override the default allocation, usually to enlarge it. If the allocation is insufficient for a logical record command, the dialog abends with a minor code of 69.
  Lrc-block-size can be any value from 1 through 32767.

- **DC PACKET SIZE IS maximum-data**
  Specifies the maximum size, in bytes, for the data stream in a QUEUE or WRITE PRINTER command. The default is 952.
  Maximum-data does not include the 72 bytes that the run-time system allocates in addition to hold system and packet-header information in the data stream. Therefore, if you accept the
default of 952, 1024 bytes will be allocated at run time for the entire data stream. The DC PACKET SIZE parameter enables you to override the default allocation, usually to enlarge it. If the allocation is insufficient for a QUEUE or WRITE PRINTER command, the dialog abends with a minor code of 19. *Maximum-data* can be any value from 1 through 32767.

### z/VSE File Characteristics Program

The z/VSE file characteristics program (IDMSFILE) enables z/VSE users to specify file characteristics at runtime in the form of control statements, providing z/VSE users with the flexibility enjoyed by z/OS users.

In the CA ADS Batch environment, you use IDMSFILE in conjunction with the CA ADS Batch runtime system (ADSBATCH) and the CA ADS Batch print log utility (ADSOBPLG). For example, to execute a CA ADS Batch application, you submit the JCL presented in [Runtime Considerations](https://docops.ca.com/display/IDMS19/Runtime+Considerations). Note that the EXEC statement executes the program IDMSFILE, and not ADSBATCH.

Control statements you supply as input to IDMSFILE describe the input, output, suspense, and log files used in the application. The last control statement specifies the program you want to execute; in this example, ADSBATCH.

#### Types of Control Statements

IDMSFILE allows you to specify four types of control statements:

- **SET OPTIONS** -- Used to define the beginning and ending columns for the control statements that follow.

- **DEFINE CHARACTERISTICS** -- Used to describe file characteristics. File characteristics include block size, record length, record format, tape label format, device type, data set organization, logical unit assignment, and printer control character format.

- **ACCEPT CHARACTERISTICS** -- Used to specify a file that contains IDMSFILE control statements. IDMSFILE processes the control statements as regular input, then continues processing the explicitly coded control statements.

- **RUN PROGRAM** -- Used to specify the program to be executed (for example, ADSBATCH) and to supply any input parameters that would, without IDMSFILE, be coded in the EXEC statement for the program.

#### Example 1

The example below illustrates the JCL and control statements you could provide to execute an CA ADS Batch application that accesses a data set whose filename is PAYROLL and a log file whose filename is ADSLOGA:

```plaintext
// DLBL SYSCTL, 'idms.sysctl',0,SD
// DLBL     ciln,'idms.cilib'
// EXTENT    ,nnnnnn
// ASSIGN    CL,SEARCH=(ciln),TEMP
// DLBL     ADSLOGA,'log.file'
```
DEFINE CHARACTERISTICS FOR FILE ADSLOGA
BLKSIZE=4096
LRECL=4092
RECFM=VARUNB
DEVTYPE=DISK ASSIGN TO SYS126.
DEFINE CHARACTERISTICS FOR FILE PAYROLL
BLKSIZE=80
LRECL=80
RECFM=FIXUNB
DEVTYPE=DISK ASSIGN TO SYS122.
DEFINE CHARACTERISTICS FOR FILE REPFILE
BLKSIZE=132
LRECL=132
RECFM=FIXUNB
DEVTYPE=DISK ASSIGN TO SYS125.
RUN PROGRAM ADBATCH PARM='DIALOG1,,runtime-parm'.
/*
DICTNAME=APPLDICT.
LOG FILE1=ADSLOGA.
DIALOG DIALOG1 OUTPUT FILENAME IS PAYROL
DIALOG STATISTICS ON FOR(DIALOG1,DIALOG2).
*/
/*
Note that the control statements for the program to be executed (in this case, ADBATCH)
immediately follow the RUN PROGRAM statement. Also note that the IDMSFILE control statements
allow you to specify file characteristics with JCL-like parameters. Alternatively, you can specify
characteristics using COBOL- or IDD-like syntax or using greatly abbreviated syntax.

Syntax rules for the four IDMSFILE control statements are described separately below. JCL for
ADBATCH is provided in Runtime Considerations (https://docops.ca.com/display/IDMS19
/Runtime+Considerations). JCL for ADSOBPLG is provided in CA ADS Batch Print Log Utility (see page 59).

Example 2

The example below illustrates the JCL and control statements you could provide to execute an CA
ADS Batch application where ADSLOGA goes to the printer and input is from cards. It also shows the
definition of the suspense, archive, and report files:

DEFINE CHARACTERISTICS FOR FILE ADSLOGA
---
// EXEC IDMSFILE,SIZE=(AUTO,128K)
@SYSFILES
@SYSCTL
// DBLB SYSIDMS,'#SYSIPT'
// EXEC IDMSFILE,SIZE=(AUTO,128K)
@SYSPARMS
/*

CONTROL CHARACTERS ARE ASA
DEVTYPE=PRINTER ASSIGN TO SYSLST.
DEFINE CHARACTERISTICS FOR FILE INFILE1
LRECL=80 RECFM=FIXUNB
SYSNO=SYSIPT
DEFINE CHARACTERISTICS FOR FILE SUSFILE
LRECL=80 BLKSIZE=4000 RECFM=FIXBLK
DEVTYPE=DISK ASSIGN TO SYS124.
DEFINE CHARACTERISTICS FOR FILE ARCFILE
LRECL=128 BLKSIZE=2564 RECFM=VARBLK
DEVTYPE=DISK ASSIGN TO SYS123.
DEFINE CHARACTERISTICS FOR FILE REPFILE
LRECL=132 BLKSIZE=132 RECFM=FIXUNB
CONTROL CHARACTERS ARE ASA
DEVTYPE=PRINTER ASSIGN TO SYSLST.
RUN PROGRAM ADSBATCH.
/
DICTNAME=APPLDICT.
LOG FILE1 = ADSLOGA NOARCHIVE.
DIALOG ARCDM INP FIL INFILE1.
DIALOG ARCDM SUS FIL SUSFILE.
DIALOG ARCD2 OUT FIL ARCFILE.
DIALOG ARCD3 OUT FIL ARCFIL.
DIALOG ARCD5 OUT FIL REPFILE.
ENTRY DIALOG ARCDM.
/*
0028
0029
0458
0459
0450
0459
0450
0450
0450
0450
0450
*/

DEFINE CHARACTERISTICS Statement

The DEFINE CHARACTERISTICS statement enables you to describe the characteristics of a file. File characteristics include block size, record length, record format, tape label format, device type, data set organization, logical unit assignment, and printer control character format. Multiple DEFINE CHARACTERISTICS statements can be coded to describe several files.

Syntax for the DEFINE CHARACTERISTICS statement is shown below:

```plaintext
  DEFINE [OVErride] CHARACTERISTICS for FILE NAME IS filename

  MAXimum BLKSZ BLKSIZE BLOCK SIZE IS blocksize CHARacters
  BLOCK CONTAINS

  MAXimum RECSIZE REcord SIZE LRECL IS record size
  RECORD CONTAINS

  logical-record-length CHARacters
```

13-Jan-2018 55/114
Note that the syntax allows you to specify characteristics in several formats, such as z/OS JCL, COBOL, and DDDL. Additionally, the syntax allows you to code abbreviated keywords.

- **DEFINE CHARACTERISTICS FOR FILE** `filename`  
  Specifies the filename of the file being described.

- **OVERRIDE**  
  Specifies that the file characteristics in the current DEFINE CHARACTERISTICS statement are to override the default characteristics of `filename`. OVERRIDE is valid only when the specified file is also named in a subsequent ACCEPT CHARACTERISTICS statement. The default characteristics of a file in an ACCEPT CHARACTERISTICS statement are:
Logical record length of 80

Block size of 80

Record format of fixed

Specify OVERRIDE if the DEFINE CHARACTERISTICS statement is overriding any of these characteristics.

MAXIMUM BLOCK SIZE IS \texttt{blocksize} CHARACTERS

Specifies the block size of the file.

Note that BLKSZ, BLKSIZE, and BLOCK SIZE are synonymous. BLOCK CONTAINS is synonymous with MAXIMUM BLOCK SIZE IS.

MAXIMUM LOGICAL RECORD LENGTH IS \texttt{lrecl} CHARACTERS

Specifies the logical record length of the file.

Note that RECSIZE, RECORD SIZE, LRECL, and LOGICAL RECORD LENGTH are synonymous. RECORD CONTAINS is synonymous with MAXIMUM LOGICAL RECORD LENGTH.

RECFM IS \texttt{FIXUNB/FIXBLK/FIXED/VARUNB/VARBLK/VARIABLE/UNDEFINED}

Specifies the record format of the file, as follows:

- \texttt{FIXUNB} specifies that the record format is fixed unblocked.
- \texttt{FIXBLK} specifies that the record format is fixed blocked.
- \texttt{FIXED LENGTH AND BLOCKED/UNBLOCKED} specifies that the record format is fixed blocked or unblocked. The default is BLOCKED.
- \texttt{VARUNB} specifies that the record format is variable unblocked.
- \texttt{VARBLK} specifies that the record format is variable blocked.
- \texttt{VARIABLE LENGTH AND BLOCKED/UNBLOCKED} specifies that the record format is variable blocked or unblocked. The default is BLOCKED.
- \texttt{UNDEFINED} specifies that the record format is undefined.

TAPE LABELS ARE OMITTED/STANDARD/NONSTANDARD/USER

Specifies the tape label format of the file. The default is STANDARD.

DEVTYPE IS DISK/TAPE/CARD/PRINTER/LIST

Specifies the device type of the file.

DSORG IS VSAM/SEQUENTIAL

Specifies the data set organization of the file. The default is SEQUENTIAL.

LU IS SYSLST/SYSPCH/SYSIPT/SYSRDR/SYS \texttt{nnn}

Specifies the logical unit assignment of the file.

CTLCHAR IS YES/ASA/NO

Specifies the printer control character format of a file assigned to a printer. The default is NO.
ACCEPT CHARACTERISTICS statement

The ACCEPT CHARACTERISTICS statement enables you to specify a file in which IDMSFILE control statements are stored. IDMSFILE processes the control statements in the specified file as regular input, then continues to process any explicitly coded control statements that follow the ACCEPT CHARACTERISTICS statement. Embedded ACCEPT CHARACTERISTICS statements are allowed.

Note that the object file can contain only IDMSFILE control statements. Control statements for the program to be executed, such as ADSBATCH, must be coded explicitly following the last IDMSFILE control statement. The object file can, however, contain a RUN PROGRAM statement.

Syntax for the ACCEPT CHARACTERISTICS statement follows:

```
Accept
CHARacteristics
filename FROM
   NAME IS =
```

- `filename`
  The name of the file that contains IDMSFILE control statements. `Filename` must be described by a DEFINE CHARACTERISTICS statement coded earlier.
  ACCEPT CHARACTERISTICS uses default file characteristics for `filename`, as follows:
  - Logical record length of 80
  - Block size of 80
  - Record format of fixed

To override any of these characteristics, specify OVERRIDE in the DEFINE CHARACTERISTICS statement that describes `filename`.

RUN PROGRAM Statement

The RUN PROGRAM statement enables you to specify the program to be executed, such as ADSBATCH or ADSOBPLG, and to specify any parameters that would, without IDMSFILE, be included in an EXEC statement for the desired program. RUN PROGRAM is the last IDMSFILE control statement you code, and is followed by control statements you code as input to the program specified in the RUN PROGRAM statement.

Syntax for the RUN PROGRAM statement is shown below:

```
RUN PROgram NAME IS =
   USING parm-list IS =
```

13-Jan-2018
RUN PROGRAM program-name
Specifies the program to be executed, such as ADSBATCH or ADSOBPLG.

USING parm-list
Specifies a parameter list that would, without IDMSFILE, be included in an EXEC statement for program-name. Parm-list must be enclosed in quotation marks if the list contains any delimiters, such as blanks, commas, semicolons, or periods.

CA ADS Batch Print Log Utility

The CA ADS Batch print log utility (ADSOBPLG) enables you to print formatted reports of selected information from a log file created during the execution of an CA ADS Batch application. The log file must contain an identifying prefix for each record; otherwise, the print log utility can print only an unformatted listing of all records in the file.

The print log utility also enables you to extract selected records from the log file without formatting. With this feature, you can, for example, extract all edit error messages from the log file in order to merge them with matching error records in the suspense file.

- How to Use the Print Log Utility (see page 59)
- DSECT for the Log File Prefix (see page 60)
- ADS Batch Control Statements (see page 61)
- z/OS JCL for ADSOBPLG (see page 63)

How to Use the Print Log Utility

In a single execution of the print log utility, you can print any or all of the following types of information contained in the log file:

- User -- Informational messages issued by WRITE TO LOG and CONTINUE commands, and by WRITE TRANSACTION commands that perform a mapout to the dialog’s output file. The message is specified in the MESSAGE TEXT parameter of the command.

- Debug -- Debugging information issued by SNAP commands and by the CA ADS Batch trace facility.

- Edit error -- Error messages issued when an input record is written to the suspense file. The runtime system writes one record consisting of the associated suspense file record number and the first 32 bytes of the input record, followed by one record for each applicable error message. In addition, if the input record was written to the suspense file by a WRITE TRANSACTION command, any message specified in the MESSAGE TEXT parameter of the command is also written to the log file as an edit error message.

- Operator -- Operator informational messages issued by WRITE TO OPERATOR commands. All messages sent to the operator are also sent to the log.
Abend -- Abnormal termination messages issued by ABORT commands and by the runtime system when an application abends. Abend messages include snap dumps.

⚠️ **Note:** If an abend occurs during a mapping operation, the snap dump goes instead to the z/OS data set associated with the ddname (linkname) IDMSSNAP or, in VSE, to SYSLST (as described under JCL (https://docops.ca.com/display/IDMS19/JCL)). An example of a mapping abend is when an input record that contains errors cannot be written to a suspense file because the MAXIMUM ERRORS for the dialog, which is specified at system generation or at runtime, has been exceeded.

Statistics -- Statistics records written by the runtime system when dialog statistics are being collected for the application.

**Selecting Log File Records**

The print log utility enables you to select log file records by:

- **Dialog** -- You can specify that only log file records created by one or more specified dialogs are included in the report.

- **User id** -- You can specify that only log file records created by specific users are included in the report. The user is determined at runtime by the current value of the $USER ($REQUESTOR) system-supplied data field; $USER ($REQUESTOR) can be initialized at runtime through an assignment command.

- **Beginning time and date** -- You can specify that only log file records created on or after a certain time and date are included in the report.

- **Ending time and date** -- You can specify that only log file records created on or before a certain time and date are included in the report.

You can specify any or all of the above selection criteria. Log file records must meet all criteria you specify in order to be included in the report.

**DSECT for the Log File Prefix**

Each record written to the log file includes an optional prefix that specifies the following information:

- The date and time the record was written

- The name of the dialog that wrote the record

- The type of record written

- The application user id

This information is used by the print log utility to select log records and format them for output. The information may also be required by reports that you create.
DSECT

The DSECT for the log file prefix, #BLPDS, is shown below:

***********************************************************************
*** BLP: BATCH LOG PREFIX DSECT
***
*** THE FOLLOWING ARE VALID VALUES FOR BLPTYPE
* 0 BLPDBUG EQU 1 DEBUG
* 1 BLPUSR EQU 2 USER MESSAGE
* 2 BLPSNSE EQU 3 EDIT ERROR RECORD
* 3 BLPWTO EQU 4 WRITE TO OPERATOR MESSAGE
* 4 BLPABND EQU 5 ABEND DUMP TEXT
* 5 BLPSTAT EQU 6 STATISTICS
* 6 BLPHITYP EQU 6 HIGHEST VALID TYPE
* 7 BLPRCMP #FLAG X'80' IF ON, LOG RECORD IS COMPRESSED
* 8 BLPCMP EQU 8
* 9 BLPULEN EQU 9 LENGTH OF USER ID
* 00 BLPLEN EQU *-BLP LENGTH OF FIXED PORTION OF PREFIX
* 01 BLPULEN DS X (CL8) LENGTH OF USER ID
* 02 BLPFLG1 DS X (CL8) FLAG BYTE
* 03 BLPFLG2 DS X (CL8) FLAG BYTE
* 04 BLPPGNAM DS CL8 PROGRAM/DIALOG NAME
* 05 BLPDATE DS CL4 DATE (00YYDDDC)
* 06 BLPTIME DS CL4 TIME (1/10000 SEC)
* 07 BLPDTK DS 0CL8 DATE/TIME (KEY)
***********************************************************************

ADS Batch Control Statements

The print log utility’s control statements enable you to specify the type of reports you want, the criteria to be applied in selecting log file records, and the information on the log file.

Syntax for the control statements is shown below:
• **MODE IS REPORT/EXTRACT**
  Specifies whether the output is formatted:
  - **REPORT** (default) specifies that the output is formatted with page headers and carriage control characters.
  - **EXTRACT** specifies that the output is unformatted. The selected records are simply extracted from the log file; each record includes the log file prefix.
    The EXTRACT option enables you, for example, to extract all EDITERR records created during the execution of an application. You can then merge these records with the suspense files that the records describe.

• **PRINT/EXCLUDE**
  Specifies whether the named record types are included in or excluded from the report. PRINT is the default.

• **ALL/(DEBUG/USER/EDITERR/OPERATOR/ABEND/STATS)**
  Specifies the types of records requested, as follows:
  - **ALL** (default) specifies all record types.
  - **DEBUG** specifies debugging information issued by SNAP commands and by the CA ADS Batch trace facility.
  - **USER** specifies messages issued by WRITE TO LOG and CONTINUE commands, and by WRITE TRANSACTION commands that perform a mapout to the terminal operator.
  - **EDITERR** specifies error messages issued when an input record is written to the suspense file.
  - **OPERATOR** specifies messages issued by WRITE TO OPERATOR commands.
  - **ABEND** specifies messages issued by the ABORT command and by the runtime system when an application abends. These messages also include snap dumps of memory produced by abends.
  - **STATS** specifies statistics written by the runtime system when dialog statistics are being collected for the application.

• **FOR ALL/DIALOGS (dialog-name)/USERIDS (user-id)**
  Specifies selection criteria by dialog and/or user id, as follows:
  - **ALL** (default) specifies that the dialog name and user id are not used as selection criteria.
- **DIALOGS (dialog-name)** specifies that only log file records created by the named dialog or dialogs are included in the report. Multiple dialog names must be separated by commas. If the DIALOGS parameter is not specified, the dialog name is not used as a selection criterion.

- **USERIDS (user-id)** specifies that only log file records created by the named user or users are included in the report. Multiple user ids must be separated by commas. If the USERIDS parameter is not specified, the user id is not used as a selection criterion.

- **FROM time-hhmmsstttt**
  Specifies that only log file records created on or after the specified time are included in the report. *Time-hhmmsstttt* specifies the time in hours (1 through 24), minutes, seconds, and tenths of seconds. Two digits of the hour must be specified; the remainder is optional. For example, 9:45 a.m. is specified as 0945. To include a date using the ON parameter, you must specify a time; you can specify the beginning of a day as 00. The default is the beginning of the day.

- **ON yyddd**
  Specifies that only log file records created on or after the specified Julian date are included in the report. The default is the date on which the print log utility is executed.

- **TO time-hhmmsstttt**
  Specifies that only log file records created on or before the specified time are included in the report. *Time-hhmmsstttt* specifies the time in hours (1 through 24), minutes, seconds, and tenths of seconds. Two digits of the hour must be specified; the remainder is optional. For example, 9:45 a.m. is specified as 0945. To include a date using the ON parameter, you must specify a time; you can specify the end of a day as 24. The default is 24, which is the end of the day.

- **ON yyddd**
  Specifies that only log file records created on or before the specified Julian date are included in the report. The default is the date on which the print log utility is executed.

- **LOG FILE IS ADSLOGA/ddname**
  Specifies the z/OS ddname or z/VSE filename of the log file. The default is ADSLOGA.

- **PREFIX/NOPREFIX**
  Specifies whether the log file contains a prefix for each record. The default is PREFIX. NOPREFIX must be specified if no prefix is included, so that the print log utility does not attempt to interpret the beginning portion of each log file record as a prefix.

---

**z/OS JCL for ADSOBPLG**

**z/OS JCL Under Central Version**

JCL for running the print log utility (ADSOBPLG) is shown below:

```plaintext
ADSOBPLG (z/OS)

//ADSOBPLG EXEC PGM=ADSOBPLG,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// /sysctl DD DSN=idms.sysctl,DISP=SHR
//adslog DD DSN=ads.log.file,DISP=SHR
//SYSLST DD SYSOUT=A
//SYSIDMS DD *
```
DMCL=dmcl-name
DICTNAME=dictionary-name
Other SYSIDMS parameters, as appropriate
/*
//SYSIPT DD *
control statements
/*
//SYSUDUMP DD SYSDOUT=A

<table>
<thead>
<tr>
<th><strong>idms.dba.</strong></th>
<th>data set name of the load library containing the DMCL and database name table load modules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>idms.loadlib</strong></td>
<td>data set name of DC/UCF load library</td>
</tr>
<tr>
<td><strong>sysctl</strong></td>
<td>ddname of SYSCTL file</td>
</tr>
<tr>
<td><strong>idms.sysctl</strong></td>
<td>data set name of SYSCTL file</td>
</tr>
<tr>
<td><strong>adslog</strong></td>
<td>ddname of CA ADS Batch log file</td>
</tr>
<tr>
<td><strong>ads.log.file</strong></td>
<td>data set name of CA ADS Batch log file</td>
</tr>
</tbody>
</table>

Local Mode

To execute ADSOBPLG in local mode, the following steps are required:

1. Remove the sysctl DD statement.

2. Add the following statements:

```plaintext
//sysjrnl DD DSN=idms.tapejrnl,DISP=NEW,UNIT=tape
//dictdb DD DSN=idms.dictdb,DISP=OLD
//dloddb DD DSN=idms.dloddb,DISP=SHR
//dmsgdb DD DSN=idms.dmsgdb,DISP=SHR
//dlogdb DD DSN=idms.dlogdb,DISP=SHR
```

additional journal file assignments, as required

<table>
<thead>
<tr>
<th><strong>sysjrnl</strong></th>
<th>ddname of journal file</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>idms.tapejrnl</strong></td>
<td>data set name of journal file</td>
</tr>
<tr>
<td><strong>tape</strong></td>
<td>symbolic device name of journal file</td>
</tr>
<tr>
<td><strong>dictdb</strong></td>
<td>ddname of data dictionary</td>
</tr>
<tr>
<td><strong>idms.dictdb</strong></td>
<td>data set name of data dictionary</td>
</tr>
<tr>
<td><strong>dloddb</strong></td>
<td>ddname of data dictionary load area</td>
</tr>
<tr>
<td><strong>idms.dloddb</strong></td>
<td>data set name of data dictionary load area</td>
</tr>
<tr>
<td><strong>dmsgdb</strong></td>
<td>ddname of data dictionary message area</td>
</tr>
<tr>
<td><strong>idms.dmsgdb</strong></td>
<td>data set name of data dictionary message area</td>
</tr>
<tr>
<td><strong>dlogdb</strong></td>
<td>ddname of data dictionary log area</td>
</tr>
<tr>
<td><strong>idms.dlogdb</strong></td>
<td>data set name of data dictionary log area</td>
</tr>
</tbody>
</table>

z/VSE JCL Under Central Version ADSOBPLG (z/VSE)

```plaintext
// DLBL SYSCTL,'idms.sysctl',1999/365,SD
// DLBL SYSIDMS,'#SYSIPT'
// DLBL ciln,'idms.cilib'
// EXTENT ,nnnnnn
// LIBDEF CL,SEARCH=(ciln),TEMP
```
// DLBL  adsloga,'log.file'
// EXTENT sysnnn,nnnnnn
ASSGN sysnnn,DISK,VOL=nnnnnn,SHR
// EXEC  IDMSFILE,SIZE=(ADSOBPLG,128K)

SYSIDMS control statements
IDMSFILE control statements
ADSOBPLG control statements

idms.sysctl  data set name of SYSCTL file
cln  filename of core-image library
idms.cilib  file-id of DC/UCF core-image library
nnnnnn  volume serial number
adsloga  filename of the log file
log.file  file-id of the log file
sysnnn  logical unit of the device on which the log file is located

SYSIDMS control statements
control statements used by SYSIDMS to describe physical runtime environments as described in CA IDMS Common Facilities Section.

IDMSFILE control statements
control statements used by IDMSFILE to describe the characteristics of the log file, as described in z/VSE File Characteristics Program (see page 53).

ADSOBPLG control statements
control statements used by ADSOBPLG

⚠️ Note: A /* card must follow each SYSIPT file.

Local Mode

To execute ADSOBPLG in local mode, the following steps are required:

1. Remove the SYSCTL DLBL card

2. Add the following statements:

   // DLBL  dictdb,'idms.dictdb',,DA
   // EXTENT sys005,nnnnnn
   // ASSGN sys005,DISK,VOL=nnnnnn,SHR
   // DLBL  dloddb,'idms.dloddb',,DA
   // EXTENT sys017,nnnnnn
   // ASSGN sys017,DISK,VOL=nnnnnn,SHR
   // DLBL  dmsgdb,'idms.dmsgdb',,DA
   // EXTENT sys016,nnnnnn
   // ASSGN sys016,DISK,VOL=nnnnnn,SHR
   // DLBL  dlogdb,'idms.dlogdb',,DA
   // EXTENT sys019,nnnnnn
   // ASSGN sys019,DISK,VOL=nnnnnn,SHR
   // TLBL  sysjrnl,'idms.tapejrnl',,nnnnnn,,f
   // ASSGN sys009,TAPE,VOL=nnnnnn
additional journal file assignments, as required
Local Mode

To execute ADSOBPLG in local mode, the following steps are required:

1. Remove the ADD-FILE-LINK statement for sysctl

2. Add the following statements:

   /ADD-FILE-LINK L-NAME=dcmsg,F-NAME=idms.sysmsg.ddldcmsg,SHARED-UPD=*YES
   /ADD-FILE-LINK L-NAME=sysjrnl,F-NAME=idms.tapejrnl

   additional journal file assignments, as required
CA ADS Batch Sample Applications

This appendix implements the following CA ADS Batch sample applications:

- **Employee-record archive application**—Archives selected employee records and their associated coverage records from an CA IDMS/DB database to an output file

- **Employee-record restore application**—Restores selected employee records and their associated coverage records from the archive file to the database

- **Employee-record report application**—Produces a report of employees by department

**Subschema**

All applications access the **employee subschema**. The subschema includes the following database records.

<table>
<thead>
<tr>
<th>Database record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>Stores employee records. The CALC key for EMPLOYEE is EMP-ID-0415.</td>
</tr>
<tr>
<td>COVERAGE</td>
<td>Stores insurance coverage records for each employee. Several COVERAGE records can be associated with an EMPLOYEE record. The relationship is described by the EMP-COVERAGE set.</td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>Stores information on the company's departments. The CALC key for DEPARTMENT is DEPT-ID-0410. Each employee belongs to a department. The relationship is described by the DEPT-EMPLOYEE set.</td>
</tr>
<tr>
<td>OFFICE</td>
<td>Stores information on the company's offices. The CALC key for OFFICE is OFFICE-CODE-0450. Each employee belongs to an office. The relationship is described by the OFFICE-EMPLOYEE set.</td>
</tr>
</tbody>
</table>

**Employee-record Archive Application**

The employee-record archive application writes selected employee records and their associated coverage records from an CA IDMS/DB database to an output file. The following diagram shows the file access, components, and flow of control for the application.
The files accessed and the dialogs used by the application are as follows.

### Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFILE1</td>
<td>Input file</td>
<td>Contains the ids of employee records to be archived</td>
</tr>
<tr>
<td>SUSFILE1</td>
<td>Suspense file</td>
<td>Contains records from INFILE1 found in error</td>
</tr>
<tr>
<td>ARCFILE</td>
<td>Output file</td>
<td>Contains archived employee records and their associated coverage records</td>
</tr>
<tr>
<td>REPFILE1</td>
<td>Output file</td>
<td>Contains a transaction summary report for the application</td>
</tr>
<tr>
<td>ADSLOGA</td>
<td>Log file</td>
<td>Contains informational and error messages produced by the application</td>
</tr>
</tbody>
</table>

### Dialogs

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCD</td>
<td>Reads input records and acts as a mainline routine, passing control to dialogs ARCD2, ARCD3, and ARCD4, as required by the application; writes erroneous input records to the suspense file</td>
</tr>
<tr>
<td>ARCD</td>
<td>Writes an employee record to the archive file, then returns control to ARCD1</td>
</tr>
<tr>
<td>ARCD</td>
<td>Writes the coverage records associated with an archived employee record to the archive file, then returns control to ARCD1</td>
</tr>
<tr>
<td>ARCD</td>
<td>Determines the transaction report lines to be written to the report output file; for each line to be written, passes control to ARCD5; returns control to ARCD1 after all lines for the transaction have been written</td>
</tr>
</tbody>
</table>
ARCD Writes a report line to the report output file, then returns control to ARCD4

Steps to create the application

To create the application, you perform the following steps:

1. Describe the input and output files in the data dictionary.
2. Describe the layouts of file, map, and work records in the data dictionary.
3. Define the file maps that associate file records with variable storage.
4. Define the process modules for the application’s dialogs.
5. Define the dialogs.

These steps are described below, followed by a discussion of executing the application.

Step 1 Describe the Files in the Data Dictionary

Each input and output file (excluding suspense and log files) must be described in the data dictionary as a file entity. You can define the files by using the IDD DDDL compiler or the IDD menu facility.

You can describe the file entities without specifying file characteristics. At runtime, under z/OS, the runtime system uses the file characteristics specified in the JCL or data set labels. Under VSE, the runtime system uses the characteristics specified by IDMSFILE control statements.

The following IDD DDDL statements add file entity descriptions for the input, archive, and report files:

ADD FILE IDD-INFILE1.
ADD FILE IDD-ARCFILE.
ADD FILE IDD-REPFILE1.

Alternatively, since you are not including file characteristics as part of the file descriptions, you can use one generic file entity to represent all three files:

ADD FILE IDD-GENERIC-FILE.

Step 2 Describe the Records in the Data Dictionary

All file, map, and work records used in the application must be described in the data dictionary. The employee-record archive application uses the following records.

<table>
<thead>
<tr>
<th>Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK-RECORD1</td>
<td>Contains miscellaneous variable fields required by the application</td>
</tr>
<tr>
<td>INPUT-RECORD1</td>
<td>Describes the layout of the input file</td>
</tr>
<tr>
<td>ARCHIVE-RECORD-EMP</td>
<td>Describes the layout of archived employee records in the archive file</td>
</tr>
</tbody>
</table>
### Record Description

<table>
<thead>
<tr>
<th>Record Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE-RECORD-COV</td>
<td>Describes the layout of archived coverage records in the archive file</td>
</tr>
<tr>
<td>REPORT-RECORD</td>
<td>Describes the layout of the report file</td>
</tr>
<tr>
<td>REPORT-HDR1</td>
<td>Describe the heading and detail lines for the report</td>
</tr>
</tbody>
</table>

You can define records by using the IDD DDDL compiler or the IDD menu facility. The record definitions are illustrated on the following pages.

#### WORK-RECORD1

**WORK-RECORD1**

```plaintext
WORK-RECORD1.
  03 WORK-DATE-YYMMDD.
    05 WORK-YY1     PICTURE IS 9(2).
    05 WORK-MM1    PICTURE IS 9(2).
    05 WORK-DD1    PICTURE IS 9(2).
  03 WORK-DATE-MMDDYY.
    05 WORK-MM2    PICTURE IS 9(2).
    05 FILLER      PICTURE IS X VALUE IS '/'.
    05 WORK-DD2    PICTURE IS 9(2).
    05 FILLER      PICTURE IS X VALUE IS '/'.
    05 WORK-YY2    PICTURE IS 9(2).
  03 WORK-LINE-CTR  PICTURE IS 99.
  03 WORK-PAGE-CTR  PICTURE IS 999.
  03 WORK-STATUS   PICTURE IS X(10).
  03 WORK-ARC-ID   PICTURE IS 9(4).
  03 WORK-ARCFILE-STATUS PICTURE IS X(3).
```

#### INPUT-RECORD1

**INPUT-RECORD1**

```plaintext
INPUT-RECORD1.
  03 INPUT-ID      PICTURE IS 9(4).
```

Include within file IDD-INFILE1.

#### ARCHIVE-RECORD-EMP

**ARCHIVE-RECORD-EMP**

```plaintext
ARCHIVE-RECORD-EMP
  03 ARCHIVE-TYPE    PICTURE IS X.
  03 ARCHIVE-DEPT-ID PICTURE IS 9(4).
  03 ARCHIVE-OFFICE-CODE PICTURE IS XXX.
  03 ARCHIVE-EMPLOYEE-RECORD PICTURE IS X(116).
```

Include within file IDD-ARCFILE.

#### ARCHIVE-RECORD-COV

**ARCHIVE-RECORD-COV**

```plaintext
ARCHIVE-RECORD-COV.
  03 ARCHIVE-TYPE    PICTURE IS X.
  03 ARCHIVE-COVERAGE-RECORD PICTURE IS X(16).
```

Include within file IDD-ARCFILE.

#### REPORT-RECORD

**REPORT-RECORD**

```plaintext
REPORT-RECORD.
  03 REPORT-CC      PICTURE IS X.
  03 REPORT-LINE    PICTURE IS X(131).
```

Include within file IDD-REPFILE1.

#### REPORT-HDR1

```plaintext
REPORT-HDR1.
03 FILLER             PICTURE IS X(15)   VALUE IS SPACES.
03 FILLER             PICTURE IS X(27)   VALUE IS 'TRANSACTION REPORT DATE:'.
03 REPORT-DATE       PICTURE IS X(8)    VALUE IS ' PAGE:'.
03 FILLER             PICTURE IS X(8)    VALUE IS ' PAGE:'.
03 REPORT-PAGE       PICTURE IS 999.

REPORT-HDR2.
03 FILLER             PICTURE IS X(15)   VALUE IS 'EMPLOYEE ID'.
03 FILLER             PICTURE IS X(28)   VALUE IS 'NAME'.
03 FILLER             PICTURE IS X(6)    VALUE IS 'STATUS'.

REPORT-DTL.
03 FILLER             PICTURE IS X(4)    VALUE IS SPACES.
03 REPORT-ID          PICTURE IS 9(4).   VALUE IS SPACES.
03 FILLER             PICTURE IS X(7)    VALUE IS SPACES.
03 REPORT-LNAME       PICTURE IS X(15).  VALUE IS SPACES.
03 FILLER             PICTURE IS X(10).
03 REPORT-FNAME       PICTURE IS X(2)    VALUE IS SPACES.
03 REPORT-STATUS      PICTURE IS X(30).

Step 3 Define the File Maps

You define a file map for each file record layout used in the application. The file maps in this
application are described below.

<table>
<thead>
<tr>
<th>Map</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP1</td>
<td>Associates record INPUT-RECORD1 with variable storage</td>
</tr>
<tr>
<td>MAP2</td>
<td>Associates record ARCHIVE-RECORD-EMP with variable storage</td>
</tr>
<tr>
<td>MAP3</td>
<td>Associates record ARCHIVE-RECORD-COV with variable storage</td>
</tr>
<tr>
<td>MAP4</td>
<td>Associates record REPORT-RECORD with variable storage</td>
</tr>
</tbody>
</table>

You define maps by using the CA IDMS/DC mapping facility. The file map definitions are illustrated below.

**MAP1**

Internal records: None
External file: INFILE1
External record: INPUT-RECORD1

**MAP2**

Internal records: DEPARTMENT
                  OFFICE
External file: ARCFIILE
Step 4 Define the Process Modules

The next step in creating the application is to define process modules consisting of Application Development System process commands. You can define process modules by using the IDD DDDL compiler or the IDD menu facility.

The process modules required by the application are presented under "Step 5: Define the Dialogs," as part of the discussion of the dialogs with which they are associated. In this way, you can see how the modules fit into the application structure.

⚠️ **Note:** The process modules shown in Step 5 have embedded comment lines, indicated by an exclamation point (!) in column 1. You do not have to key in these lines.

Step 5 Define the Dialogs

The next step in defining the application is to define its dialogs. A dialog is a collection of application components created in earlier steps, including file maps and process modules. You can define a dialog by using the online dialog compiler.

The dialogs in the archive application are described below.
Dialog Description

ARCD  Reads input records and acts as a mainline routine, passing control to ARCD2, ARCD3, and ARCD4, as required by the application; writes erroneous input records to the suspense file.

ARCD  Writes the specified employee record to the archive file, then returns control to ARCD1.

ARCD  Writes the coverage records associated with the specified employee record to the archive file, then returns control to ARCD1.

ARCD  Determines the transaction report lines to be written to the report output file; for each line to be written, passes control to ARCD5; returns control to ARCD1 once all lines have been written for the transaction.

ARCD  Writes a report line to the report output file, then returns control to ARCD4.

The definitions for these dialogs are provided separately below, along with the process modules associated with the dialogs and the mapin or mapout operations that the dialogs perform at runtime. For an illustration of how these dialogs fit together, see the diagram earlier in this section.

Dialog ARCD1

Dialog ARCD1 executes at the beginning of the application. ARCD1 reads an input file that contains a list of ids of employee records to be archived. ARCD1 serves as a mainline dialog, passing control to other dialogs, as required, to archive the employee and associated coverage records and to write report lines to a report file. ARCD1 also writes erroneous input records to a suspense file.

The dialog definition for ARCD1 is illustrated below:
The premap process, mapin operation, and response process are shown below.

**Dialog ARCD1: Premap Process**

```plaintext
!**********************************************************************
!*ARCD1-PM                                                            *
!*  -EXECUTED ONCE AT THE BEGINNING OF THE APPLICATION WHEN           *
!*   DLG ARCD1 BEGINS EXECUTION.                                      *
!*  -PERFORMS APPLICATION INITIALIZATION, THEN READS THE FIRST        *
!*   INPUT RECORD.                                                    *
!*  READS ONLY WHEN THE WORK RECORD SUBSCHEMA NAME IS EMPSS01.        *
!**********************************************************************
```

**Mapin**

- **Map name:** MAP1
- **Ddname:** INFILE1
- **Suspense ddname:** SUSFILE1

**Response process**

**ARCD1-RESPONSE**

*default RP*

**Work records**

- WORK-RECORD1
- REPORT-HDR1
- REPORT-DTL

**Subschema**

- EMPSS01

IDMSDB--Dialog ARCD1
CALL INIT.
READ TRANSACTION.
!
!
!*****************************************************************************
!*SUBROUTINE INIT              *
!*-SET UP FOR TRANSACTION REPORT.    *
!*****************************************************************************
DEFINE INIT.
  MOVE DATE TO WORK-DATE-YYMMDD.
  MOVE WORK-YY1 TO WORK-YY2.
  MOVE WORK-MM1 TO WORK-MM2.
  MOVE WORK-DD1 TO WORK-DD2.
  MOVE WORK-DATE-MMDDYY TO REPORT-DATE.
  MOVE 55 TO WORK-LINE-CTR.
  GOBACK.

Dialog ARCD1: Mapin Operation

External field          Internal field
INPUT-ID.......................>INPUT-ID

Dialog ARCD1: Response Process

!*****************************************************************************
!*ARCD1-RESPONSE        *
!* -EXECUTED AFTER DLG ARCD1'S MAPIN OPERATION. DEFAULT RESPONSE    *
!* PROCESS FOR ARCD1.     *
!* -PERFORMS MAINLINE PROCESSING OF INPUT RECORD.                     *
!*****************************************************************************
READY USAGE-MODE UPDATE.
!
!
!*****************************************************************************
!*-TERMINATE APPLICATION ON AN EOF CONDITION *
!*****************************************************************************
IF $EOF
  DO.
    WRITE LOG TEXT '***EOF ON INPUT***'.
    LEAVE APPLICATION.
  END.
!
!
!*****************************************************************************
!*-INPUT-ID CONTAINS THE ID OF THE EMP REC  *
!* TO BE ARCHIVED.               *
!*-ATTEMPT TO RETRIEVE THE RECORD FROM THE  *
!* DATABASE.                     *
!*-IF THE RECORD DOES NOT EXIST, CALL AN  *
!* ERROR ROUTINE.                *
!*-ON ANY OTHER DB ERROR, TERMINATE THE APP.   *
!*****************************************************************************
MOVE INPUT-ID TO EMP-ID-0415.
OBTAIN CALC EMPLOYEE.
IF DB-REC-NOT-FOUND
  CALL ERRRTN.
IF DB-ANY-ERROR
  ABORT TEXT 'DB ERROR ON EMPLOYEE OBTAIN'.
!
!
!*****************************************************************************
!*-PASS CONTROL TO DLG ARCD2, WHICH ARCHIVES  *
!* THE EMPLOYEE RECORD.             *
!*-PASS CONTROL TO DLG ARCD3, WHICH ARCHIVES  *
!* THE ASSOCIATED COVERAGE RECORDS.   *
!*****************************************************************************
LINK NOSAVE 'ARC2'.
LINE NOSAVE 'ARC3'.
!
Dialog ARCD2

Dialog ARCD2 executes when it receives control from ARCD1. ARCD2 archives an employee record.

The dialog definition for ARCD2 is illustrated below:
The premap process and mapout operation are shown below,

**Dialog ARCD2: Premap Process**

```
*---------------------------------------------------------------------------*
!* ARCD2-PM                                                            *
!* -EXECUTED AT THE BEGINNING OF DLG ARCD2.                               *
!* -ARCHIVES AN EMPLOYEE RECORD.                                         *
!* -THE EMP Archive REC CONSISTS OF THE FOLLOWING FIELDS:                *
!*   -TYPE FIELD (INTERNAL FIELD IS $RESPONSE)                          *
!*   -DEPARTMENT ID                                                     *
!*   -OFFICE ID                                                         *
!*   -EMPLOYEE RECORD                                                   *
*---------------------------------------------------------------------------*

*---------------------------------------------------------------------------*
!* -MOVE RECORD TYPE TO $RESPONSE.                                       *
*---------------------------------------------------------------------------*
MOVE 'E' TO $RESPONSE.

*---------------------------------------------------------------------------*
!* -RETRIEVE THE EMPLOYEE'S OFFICE AND                                    *
```
!* DEPARTMENT IDS. *!
!*-ON ANY DATABASE ERROR, ABORT APPL. *!
!*********************************************************
OBTAIN OWNER WITHIN OFFICE-EMPLOYEE.
IF DB-ANY-ERROR
   ABORT TEXT 'DB ERROR ON OFFICE OBTAIN'.
OBTAIN OWNER WITHIN DEPT-EMPLOYEE.
IF DB-ANY-ERROR
   ABORT TEXT 'DB ERROR ON DEPARTMENT OBTAIN'.
!
!*********************************************************
!*-MOVE EMPLOYEE RECORD TO ARCHIVE RECORD *!
!*********************************************************
MOVE EMPLOYEE TO ARCHIVE-EMPLOYEE-RECORD.
!
!*********************************************************
!*-WRITE THE RECORD, THEN RETURN TO ARCD1. *!
!*********************************************************
WRITE TRANSACTION RETURN.

**Dialog ARCD2: Mapout Operation**

<table>
<thead>
<tr>
<th>Internal field</th>
<th>External field</th>
</tr>
</thead>
<tbody>
<tr>
<td>$RESPONSE ................</td>
<td>&gt;ARCHIVE-TYPE</td>
</tr>
<tr>
<td>DEPT-ID-0410 ...........</td>
<td>&gt;ARCHIVE-DEPT-ID</td>
</tr>
<tr>
<td>OFFICE-CODE-0450 ........</td>
<td>&gt;ARCHIVE-OFFICE-CODE</td>
</tr>
<tr>
<td>ARCHIVE-EMPLOYEE-RECORD.</td>
<td>&gt;ARCHIVE-EMPLOYEE-RECORD</td>
</tr>
</tbody>
</table>

**Dialog ARCD3**

Dialog ARCD3 executes when it receives control from ARCD1. ARCD3 archives all coverage records associated with the employee record being archived.

The dialog definition for ARCD3 is illustrated below:
The premap process and mapout operation are shown below.

**Dialog ARCD3: Premap Process**

```plaintext
![ARCD3-PM]

* -EXECUTED AT THE BEGINNING OF DLG ARCD3.
* -ARCHIVES ALL ASSOCIATED COVERAGE RECORDS.
* -THE COVERAGE ARCHIVE REC CONSISTS OF THE FOLLOWING FIELDS:
  * -TYPE FIELD (INTERNAL FIELD IS $RESPONSE)
  * -COVERAGE RECORD

---

**Mapout**

Map name: MAP3
Ddname: ARCFI

---

**Work records**

None

---

**Subschema**

EMPSS01

---

IDMSDB--Dialog ARCD3
```
![Image](https://example.com/image.png)

**Dialog ARCD3: Mapout Operation**

<table>
<thead>
<tr>
<th>Internal field</th>
<th>External field</th>
</tr>
</thead>
<tbody>
<tr>
<td>$RESPONSE..................&gt;ARCHIVE-TYPE</td>
<td>ARCHIVE-COVERAGE-RECORD........&gt;ARCHIVE-COVERAGE-RECORD</td>
</tr>
</tbody>
</table>

**Dialog ARCD4**

Dialog ARCD4 executes when it receives control from ARCD1. ARCD4 prepares report lines to be written to the report file and passes control to ARCD5, which actually writes the lines to the output file.

The dialog definition for ARCD4 is illustrated below:

![Diagram](https://example.com/diagram.png)

IDMSDB--Dialog ARCD4
The premap process is shown below.

**Dialog ARCD4: Premap Process**

```
!******************************************************************************
!*ARCD4-PM
!* -EXECUTED AT THE BEGINNING OF DLG ARCD4.
!* -DETERMINES REPORT LINE(S) TO BE WRITTEN.
!* -THE REPORT RECORD CONSISTS OF THE FOLLOWING FIELDS:
!*     -REPORT-CC  CARRIAGE CONTROL CHARACTER
!*     -REPORT-LINE  ACTUAL REPORT LINE
!* -THIS PROCESS MOVES THE PROPER DATA INTO THESE FIELDS, THEN
!* PASSES CONTROL TO ARCD5, WHICH WRITES A LINE.
!*******************************************************************************

ADD 1 TO WORK-LINE-CTR.

!*************************************************
!*-IF LINE COUNTER > 55, CALL A ROUTINE THAT
!* PRINTS HEADING LINES, THEN MOVE 0 TO
!* REPORT-CC, SO THAT THE FIRST DETAIL LINE
!* IS DOUBLE SPACED.
!* -IF LINE COUNTER IS NOT > 55, MOVE ' ' TO
!* REPORT-CC, SO THAT THE DETAIL LINE IS
!* SINGLE SPACED.
!*******************************************************************************

IF WORK-LINE-CTR > 55
DO.
CALL PRINTHDR.
MOVE '0' TO REPORT-CC.
END.
ELSE
MOVE ' ' TO REPORT-CC.
!
!*************************************************
!*-MOVE THE DETAIL LINE TO REPORT-LINE.
!*-PASS CONTROL TO ARCD5, WHICH WRITES THE
!* LINE.
!*-RETURN CONTROL TO ARCD1.
!*******************************************************************************

MOVE REPORT-DTL TO REPORT-LINE.
LINK NOSAVE 'ARCD5'.
RETURN.
!
!*************************************************
!*SUBROUTINE PRINTHDR
!*-CALLED WHEN PAGE HEADERS ARE TO BE PRINTED
!*-SET UP FIRST HEADING LINE, THEN PASS
!* CONTROL TO ARCD5, WHICH Writes IT.
!*-SET UP SECOND HEADING LINE, THEN PASS
!* CONTROL TO ARCD5, WHICH Writes IT.
!*-GO BACK TO MAIN ROUTINE TO WRITE THE
!* DETAIL LINE.
!*******************************************************************************

DEFINE PRINTHDR.
MOVE 1 TO WORK-LINE-CTR.
ADD 1 TO WORK-PAGE-CTR.
MOVE WORK-PAGE-CTR TO REPORT-PAGE.
MOVE REPORT-HDR1 TO REPORT-LINE.
MOVE '1' TO REPORT-CC.
LINK NOSAVE 'ARCD5'.
MOVE REPORT-HDR2 TO REPORT-LINE.
MOVE '-' TO REPORT-CC.
LINK NOSAVE 'ARCD5'.
GOBACK.
```
Dialog ARCD5

Dialog ARCD5 executes when it receives control from ARCD4. ARCD5 writes a report line to the report file.

The dialog definition for ARCD5 is illustrated below:

**Premap process**

**ARCD5-PM**

**Mapout**

**Map name:** MAP4  
**Ddname:** REPFILE1

**Work records**

**None**

**Subschema**

**EMPSS01**

**IDMSDB--Dialog ARCD5**

The premap process and mapout operation are shown below.

**Dialog ARCD5: Premap Process**

```
!**********************************************************************
!*ARCD5-PM
!* -EXECUTED AT THE BEGINNING OF DLG ARCD5.
!* -WRITES A REPORT LINE, THEN PASSES CONTROL BACK TO ARCD4.
* WRITES A REPORT LINE, THEN PASSES CONTROL BACK TO ARCD4.
* WRITE TRANSACTION RETURN.
**********************************************************************
```

**Dialog ARCD5: Mapout Operation**
Executing the Application

You execute the application by executing the batch program ADSBATCH, as described in Runtime Considerations (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

Physical File Characteristics

Characteristics of the physical files that the application accesses are shown below. Under z/OS, the JCL specifications or data set label for each file should specify the record formats, logical record lengths, and block sizes indicated. Under z/VSE, these characteristics should be specified in IDMSFILE control statements. The JCL for executing an CA ADS Batch application is provided in Runtime Considerations (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

<table>
<thead>
<tr>
<th>Data set name</th>
<th>DDNAME/ filename</th>
<th>Record format</th>
<th>Logical record size</th>
<th>Block size</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFILE1</td>
<td>INFILE1</td>
<td>FB</td>
<td>4</td>
<td>4000</td>
</tr>
<tr>
<td>SUSFILE1</td>
<td>SUSFILE1</td>
<td>FB</td>
<td>4</td>
<td>4000</td>
</tr>
<tr>
<td>ARCF</td>
<td>E</td>
<td>ARCF</td>
<td>E</td>
<td>VB</td>
</tr>
<tr>
<td>REPFILE1</td>
<td>REPFILE1</td>
<td>FB</td>
<td>133</td>
<td>1330</td>
</tr>
<tr>
<td>ADSLOGA</td>
<td>ADSLOGA</td>
<td>VB</td>
<td>320</td>
<td>12804</td>
</tr>
</tbody>
</table>

Control statements

The JCL can also include control statements. In the following sample set of control statements, you specify the application entry point, the requestor’s id, and the ddname of the log file:

ENTRY POINT DIALOG ARCD1.
REQUESTOR MPK.
LOG FILE1= ADSLOGA.

The contents of INFILE1 before the application is executed, and the contents of REPFILE1, SUSFILE1, ARCF|E, and ADSLOGA after the application has been executed are shown below:

**Note:** The employee-record restore application, implemented later in this appendix, assumes that the archive file is ordered by employee id. Therefore, make sure that your input file for this application is also ordered by employee id.
**Employee-Record Restore Application**

The employee-record restore application restores selected employee records and their associated coverage records from an archive file (created in the employee-record archive application) to an CA IDMS/DB database. The following drawing shows the file access, components, and flow of control for the application.
The files accessed by this application are the same as those accessed in the archive application. ARCFILE, which was an output file in the archive application, is an input file in the restore application.

The restore application uses a global application structure defined with the application generator. In the drawing above, functions are represented by dotted rectangles; responses are represented by circles.

The application structure enables the application to access an input file that has more than one record layout (ARCFILE). The structure ensures that type E (employee) records are mapped in by dialog RESD2, and that type C (coverage) records are mapped in by dialog RESD3.

Processing

At runtime, the following processing occurs when RESD2 or RESD3 reads a record from the archive file:

- **If RESD2 reads a type E record**, the record is mapped in using MAP2 and a response process is selected.

- **If RESD2 reads a type C record**, control passes immediately to dialog RESD3. The record is not mapped in; RESD3 must perform a read operation to map in the record.

- **If RESD3 reads a type C record**, the record is mapped in using MAP3 and a response process is selected.
• If RESD3 reads a type E record, control passes immediately to dialog RESD2. The record is not mapped in; RESD2 must perform a read operation to map in the record.

Dialogs Used

The dialogs used in the application are described below.

<table>
<thead>
<tr>
<th>Dialog Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESD1 Reads input records and acts as a mainline routine, passing control (by means of the application structure) to dialogs RESD2 and ARCD4, as required by the application; writes erroneous input records to a suspense file.</td>
</tr>
<tr>
<td>RESD2 Finds the requested employee record in the archive file; restores the record; performs another read operation to begin reading associated coverage records; if a coverage record is read, control passes automatically to dialog RESD3; returns control to RESD1 when all associated coverage records have been restored.</td>
</tr>
<tr>
<td>RESD3 Restores coverage records associated with the restored employee record; if an employee record is read, control passes automatically back to dialog RESD2.</td>
</tr>
<tr>
<td>ARCD 4 and ARCD 5 Write transaction report lines to an output file.</td>
</tr>
</tbody>
</table>

Steps

To create the application, you perform the following steps:

1. Define the application structure.
2. Define the process modules for the application’s dialogs.
3. Define the dialogs.

Note that you do not define the files, records, and file maps used by the application, nor do you define dialogs ARCD4 and ARCD5; you already defined these components as part of the archive application.

The steps you perform to define the restore application are described on the following pages, followed by a discussion of executing the application.

**Step 1 Define the Application Structure**

You define the application structure by using the online application generator. The application structure consists of responses, functions, and a task code. The specifications you make are shown below:

**Application Specifications**
## Component Name Characteristics

<table>
<thead>
<tr>
<th>Responses</th>
<th>R2</th>
<th>Invokes F2 LINK command NOSAVE option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R4</td>
<td>Invokes F4 LINK command NOSAVE option</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Invokes F2 TRANSFER command NOFINISH option</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Invokes F3 TRANSFER command NOFINISH option</td>
</tr>
<tr>
<td>Functions</td>
<td>F1</td>
<td>Assoc. with RESD1 Valid responses: F1 F4</td>
</tr>
<tr>
<td></td>
<td>F2</td>
<td>Assoc. with RESD2 Valid responses: E C</td>
</tr>
<tr>
<td></td>
<td>F3</td>
<td>Assoc. with RESD3 Valid responses: E C</td>
</tr>
<tr>
<td></td>
<td>F4</td>
<td>Assoc. with ARCD4</td>
</tr>
</tbody>
</table>

### Task code

| RESAPPL | Invokes F1 |

### Note:

For responses, you provide the NOSAVE and NOFINISH specifications.

---

## Step 2 Define the Process Modules

As the next step in defining the employee-record restore application, you define process modules consisting of Application Development System process commands. You can define process modules using the IDD DDDL compiler or the IDD menu facility.

The process modules required by the application are presented under "Step 3: Define the Dialogs," together with the dialogs with which they are associated. In this way, you can see how the modules fit into the application structure.
Step 3 Define the Dialogs

The next step in defining the application is to define its dialogs. A dialog is a collection of application components created in earlier steps, including file maps and process modules. You can define a dialog by using the online dialog compiler.

Dialogs Used

The dialogs in the restore application are described below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESD</td>
<td>Reads input records and acts as a mainline routine, passing control (by means of the application structure) to dialogs RESD2 and ARCD4, as required by the application; writes erroneous input records to a suspense file</td>
</tr>
<tr>
<td>RESD</td>
<td>Finds the requested employee record in the archive file; restores the record; performs another read operation to begin reading associated coverage records; if a coverage record is read, control passes automatically to dialog RESD3; returns control to RESD1 when all associated coverage records have been restored</td>
</tr>
<tr>
<td>RESD</td>
<td>Restores coverage records associated with the restored employee record; if an employee record is read, control passes automatically back to dialog RESD2</td>
</tr>
<tr>
<td>ARCD</td>
<td>Write transaction report lines to an output file</td>
</tr>
</tbody>
</table>

The definitions for these dialogs (excluding ARCD4 and ARCD5, which were defined in the employee-record archive application) are provided separately below, along with the process modules associated with the dialogs and the mapin or mapout operations that these dialogs perform at runtime. For an illustration of how these dialogs fit together, see the diagram earlier in this section.

Dialog RESD1

Dialog RESD1 reads input records and acts as a mainline routine, passing control (by means of the application structure) to dialogs RESD2 and ARCD4, as required by the application. RESD1 also writes erroneous input records to a suspense file.

The dialog definition for RESD1 is illustrated below:
The premap process, mapin operation, and response process are shown below.

Dialog RESD1: Premap Process
CA IDMS - 19.0

!**********************************************************************
!*RESD1-PM
!* -EXECUTED AT THE BEGINNING OF THE APPLICATION WHEN
!*  DLG RESD1 BEGINS EXECUTION.
!* -PERFORMS APPLICATION INITIALIZATION, THEN READS THE FIRST
!*  INPUT RECORD.
!* **********************************************************************
CALL INIT.
READ TRANSACTION.
!

!*************************************************
!*SUBROUTINE INIT
!*-INITIALIZE WORK-ARC-ID, WHICH CONTAINS
!* THE ID OF THE MOST RECENTLY READ ARCHIVED
!* EMPLOYEE RECORD.
!* -SET UP FOR TRANSACTION REPORT.
!*************************************************
DEFINE INIT.
MOVE 0 TO WORK-ARC-ID.
MOVE DATE TO WORK-DATE-YYMMDD.
MOVE WORK-YY1 TO WORK-YY2.
MOVE WORK-MM1 TO WORK-MM2.
MOVE WORK-DD1 TO WORK-DD2.
MOVE WORK-DATE-MMDDYY TO REPORT-DATE.
MOVE 55 TO WORK-LINE-CTR.
GOBACK.

Dialog RESD1: Mapin Operation
External field Internal field
INPUT-ID.................>INPUT-ID

Dialog RESD1: Response Process

!**********************************************************************
!*RESD1-RESPONSE
!* -EXECUTED AFTER DLG RESD1'S MAPIN OPERATION. DEFAULT RESPONSE
!* PROCESS FOR RESD1.
!* -PERFORMS MAINLINE PROCESSING OF INPUT RECORD.
!* **********************************************************************
READY USAGE-MODE UPDATE.
!

!*************************************************
!*-TERMINATE APPLICATION ON AN EOF CONDITION
!*************************************************
IF $EOF
  DO.
  WRITE LOG TEXT '***EOF ON INPUT***'.
  LEAVE APPLICATION.
  END.
!

!*************************************************
!*-INPUT-ID CONTAINS THE ID OF THE EMP REC
!* TO BE RESTORED.
!*-ATTEMPT TO RETRIEVE THE RECORD FROM THE
!* DATABASE.
!*-IF THE RECORD ALREADY EXISTS, CALL AN
!* ERROR ROUTINE.
!*-ON ANY OTHER ERROR, TERMINATE THE APP.
!*************************************************
MOVE INPUT-ID TO EMP-ID-0415.
OBTAIN CALC EMPLOYEE.
IF DB-STATUS-OK
  CALL ERRRTN.
  IF DB-REC-NOT-FOUND
    NEXT COMMAND.
ELSE
   IF DB-ANY-ERROR
      ABORT TEXT 'DB ERROR ON EMPLOYEE OBTAIN'.
   !
!
*******************************************************************************
!*-WORK-ARCFILE-STATUS IS SET TO EOF WHEN *
!* THE ARCHIVE FILE HAS REACHED THE EOF. *
!*-NOTFND IS A SUBROUTINE THAT IS CALLED *
!* WHEN THE EMP REC TO BE RESTORED CANNOT BE *
!* FOUND IN THE ARCHIVE FILE. *
!* IF WORK-ARCFILE-STATUS = EOF, CALL NOTFND. *
*******************************************************************************
IF WORK-ARCFILE-STATUS = 'EOF'
   CALL NOTFND.
!
!
*******************************************************************************
!*-PASS CONTROL TO FUNCTION F2, WHICH *
!* EXECUTES DIALOG RESD2. *
!*-F2 (RESD2) AND F3 (RESD3), WHICH IS A VALID *
!* FUNCTION FROM F2, READ THE ARCHIVE FILE *
!* AND ATTEMPT TO RESTORE THE EMP REC AND ITS *
!* ASSOCIATED COVERAGE RECS. IF THE EMP REC *
!* CANNOT BE FOUND, THESE DIALOGS SET WORK- *
!* STATUS TO 'NOT FOUND'. *
*******************************************************************************
MOVE 'R2' TO AGR-CURRENT-RESPONSE.
MOVE SPACES TO WORK-STATUS.
EXECUTE NEXT FUNCTION.
!
!
*******************************************************************************
!*-IF THE REQUESTED EMP REC WAS NOT FOUND, *
!* CALL THE NOTFND ERROR ROUTINE. *
!*-IF THE RECORD WAS FOUND AND RESTORED, *
!* CALL THE FOUND ROUTINE. *
*******************************************************************************
IF WORK-STATUS = 'NOT FOUND'
   CALL NOTFND.
ELSE
   CALL FOUND.
!
*******************************************************************************
!*SUBROUTINE ERRRTN *
!*-CALLED WHEN THE REQUESTED EMP REC IS *
!* ALREADY ON THE DATABASE. *
!*-SET UP FOR TRANSACTION REPORT, THEN *
!* PASS CONTROL TO FUNCTION F4 (DLG ARCD4), *
!* WHICH, ALONG WITH ARCD5, WRITES A REPORT *
!* LINE. *
!*-SET INPUT-ID OF THE INPUT MAP IN ERROR. *
!*-ISSUE WRITE TRANSACTION COMMAND, WHICH *
!* WRITES THE INPUT RECORD TO THE SUSPENSE *
!* FILE, THEN READS THE NEXT INPUT RECORD. *
*******************************************************************************
DEFINE ERRRTN.
   MOVE 'EMPLOYEE ALREADY ON DATABASE' TO REPORT-STATUS.
   MOVE INPUT-ID TO REPORT-ID.
   MOVE EMP-LAST-NAME-0415 TO REPORT-LNAME.
   MOVE EMP-FIRST-NAME-0415 TO REPORT-FNAME.
   MOVE 'R4' TO AGR-CURRENT-RESPONSE.
   EXECUTE NEXT FUNCTION.
   MODIFY MAP TEMP FOR (INPUT-ID) EDIT ERROR.
   WRITE TRANSACTION.
!
!
*******************************************************************************
!*SUBROUTINE NOTFND *
!*-CALLED WHEN THE REQUESTED EMP REC IS

!* NOT ON THE ARCHIVE FILE. (NOTE THAT THE      *
!* APPL. ASSUMES THAT THE INPUT FILE AND       *
!* ARCHIVE FILE ARE BOTH ORDERED BY EMP ID.)    *
!* SET UP FOR TRANSACTION REPORT, THEN        *
!* PASS CONTROL TO FUNCTION F4 (DLG ARCD4),    *
!* WHICH, ALONG WITH ARCD5, WRITES A REPORT    *
!* LINE.                                      *
!* SET INPUT-ID OF THE INPUT MAP IN ERROR.     *
!* ISSUE WRITE TRANSACTION COMMAND, WHICH       *
!* WRITES THE INPUT RECORD TO THE SUSPENSE     *
!* FILE, THEN READS THE NEXT INPUT RECORD.     *
******************************************************************************
DEFINE NOTFND.
MOVE 'EMPLOYEE NOT FOUND' TO REPORT-STATUS.
MOVE INPUT-ID TO REPORT-ID.
MOVE SPACES TO REPORT-LNAME.
MOVE SPACES TO REPORT-FNAME.
MOVE 'R4' TO AGR-CURRENT-RESPONSE.
EXECUTE NEXT FUNCTION.
MODIFY MAP TEMP FOR (INPUT-ID) EDIT ERROR.
WRITE TRANSACTION.
!
!*******************************************************************************
!*SUBROUTINE FOUND            *
!*-CALLED WHEN THE REQUESTED EMP REC HAS BEEN *
!* FOUND ON THE ARCHIVE FILE AND RESTORED.     *
!* SET UP FOR TRANSACTION REPORT, THEN        *
!* PASS CONTROL TO FUNCTION F4 (DLG ARCD4),    *
!* WHICH, ALONG WITH ARCD5, WRITES A REPORT    *
!* LINE.                                      *
!* READ THE NEXT INPUT RECORD.                *
******************************************************************************
DEFINE FOUND.
MOVE 'EMPLOYEE RESTORED' TO REPORT-STATUS.
MOVE INPUT-ID TO REPORT-ID.
MOVE EMP-LAST-NAME-0415 TO REPORT-LNAME.
MOVE EMP-FIRST-NAME-0415 TO REPORT-FNAME.
MOVE 'R4' TO AGR-CURRENT-RESPONSE.
EXECUTE NEXT FUNCTION.
READ TRANSACTION.

Dialog RESD2

Dialog RESD2 processes archived employee records. RESD2 finds the requested employee record in
the archive file and restores it. It then performs another read operation to begin reading associated
coverage records; if a coverage record is read, control passes automatically to dialog RESD3.

The dialog definition for RESD2 is illustrated below:
The premap process, mapin operation, and response process are shown below.

**Dialog RESD2: Premap Process**

```plaintext
!*RESP2-PM
!* -EXECUTED AT THE BEGINNING OF DLG RESD2.
!* -IN THIS APPLICATION, CONTROL PASSES HERE WHEN:
!*   -RESD1 PASSES CONTROL TO RESD2 IN ORDER TO FIND AND
!*   RESTORE THE REQUESTED EMPLOYEE RECORD.
!*   -RESPD2-RESPONSE ISSUES A CONTINUE COMMAND AFTER AN
```
EMPLOYEE RECORD HAS BEEN READ SO THAT RESD2-PM CAN PROCESS THE RECORD.

-DLG RESD3 IS TO READ THE NEXT RECORD FROM THE ARCHIVE FILE, BUT THE NEXT RECORD IS AN EMPLOYEE RECORD.

IN THIS CASE, CONTROL PASSES AUTOMATICALLY BACK TO RESD2, AS SPECIFIED BY THE APPLICATION DEFINITION.

-RESD2-PM PROCESSES THE MOST RECENTLY READ ARCHIVE FILE.

-NOTE THAT THE PREMAP PROCESS ASSUMES THAT BOTH THE INPUT AND ARCHIVE FILE RECORDS ARE ORDERED BY EMP ID.

-NOTE THAT WORK-STATUS IS SET TO 'RESTORED' WHEN THE REQUESTED EMP HAS BEEN RESTORED ONTO THE DATABASE.

READY USAGE-MODE UPDATE.

!!-IF THE ARCHIVED RECORD HAS BEEN RESTORED, RETURN TO RESD1 TO READ THE NEXT INPUT RECORD.

IF WORK-STATUS = 'RESTORED'
RETURN.

!!-WORK-ARC-ID IS THE ID OF THE MOST RECENTLY READ ARCHIVED EMP REC.

INPUT-ID IS THE ID OF THE EMP-REC TO BE RESTORED.

IF THERE IS A MATCH, RESTORE THE EMP REC, MOVE 'RESTORED' TO WORK-STATUS, THEN READ THE NEXT ARCHIVE FILE RECORD IN ORDER TO RESTORE THE EMPLOYEE'S ASSOCIATED COVERAGE RECORDS.

IF WORK-ARC-ID = INPUT-ID
DO.
CALL RESTRTN.
MOVE 'RESTORED' TO WORK-STATUS.
READ TRANSACTION.
END.

!!-WORK-ARC-ID > INPUT ID MEANS THAT NO MATCH WAS FOUND.

IN THIS CASE, MOVE 'NOT FOUND' TO WORK-STATUS, AND RETURN TO RESD1.

IF WORK-ARC-ID > INPUT-ID
DO.
MOVE 'NOT FOUND' TO WORK-STATUS.
RETURN.
END.

!!-WORK-ARC-ID < INPUT ID (THE ONLY REMAINING POSSIBILITY) MEANS THAT THE ARCHIVE FILE SHOULD BE READ UNTIL A MATCH IS FOUND OR UNTIL THE ARCHIVE EMP ID IS GREATER THAN THE INPUT EMP ID.

READ TRANSACTION.

!!SUBROUTINE RESTRTN

-CALLED WHEN A MATCH IS FOUND.

-SET CURRENCY ON EMPLOYEE'S OFFICE AND DEPARTMENT, WHOSE IDS WERE ARCHIVED ALONG
WITH THE RECORD.
-ON ANY DATABASE ERROR, ABORT THE APPL.
-RESTORE THE ARCHIVED EMPLOYEE RECORD ONTO
THE DATABASE.

DEFINE RESTRTN.
FIND CALC OFFICE.
IF DB-ANY-ERROR
   ABORT TEXT '***DB ERROR ON FIND OFFICE***'.
FIND CALC DEPARTMENT.
IF DB-ANY-ERROR
   ABORT TEXT '***DB ERROR ON FIND DEPARTMENT***'.
STORE EMPLOYEE.
IF DB-ANY-ERROR
   ABORT TEXT '***DB ERROR ON STORE EMPLOYEE***'.
GOBACK.

Dialog RESD2: Mapin Operation

External field Internal field

ARCHIVE-TYPE...............>$RESPONSE
ARCHIVE-DEPT-ID.............>DEPT-ID-0410
ARCHIVE-OFFICE-CODE........>OFFICE-CODE-0450
ARCHIVE-EMPLOYEE-RECORD.....>$ARCHIVE-EMPLOYEE-RECORD

Dialog RESD2: Response Process

!*RESD2-RESPONSE
!* -EXECUTED AFTER AN EMPLOYEE RECORD HAS BEEN READ FROM THE
!*  ARCHIVE FILE, OR AFTER AN ARCHIVE FILE EOF. RESD2’S DEFAULT
!*  RESPONSE PROCESS.
!* -PASSES CONTROL AS APPROPRIATE, DEPENDING ON CURRENT
!*  CONDITIONS.
!*---------------------------------------------------------------------
!
!*---------------------------------------------------------------------
!*-ON ARCHIVE FILE EOF, MOVE EOF TO WORK-ARCFILE-STATUS.
!*-IF THE REQUESTED EMPLOYEE HAS NOT BEEN RE-
!*  STORED, MOVE ‘NOT FOUND’ TO WORK-STATUS
!*-RETURN TO RESD1.
!*---------------------------------------------------------------------
IF $EOF
   DO.
      MOVE ‘EOF’ TO WORK-ARCFILE-STATUS.
      IF WORK-STATUS NE ‘RESTORED’
         MOVE ‘NOT FOUND’ TO WORK-STATUS.
   ELSE
      NEXT.
   RETURN.
END.

!*---------------------------------------------------------------------
!*-MOVE THE ARCHIVED RECORD TO THE EMPLOYEE DATABASE RECORD.
!*-MOVE THE EMP ID TO WORK-ARC-ID (ID OF THE
!*  MOST RECENTLY READ ARCHIVED EMP REC).
!*-REEXECUTE THE PREMAP PROCESS TO PROCESS
!*  THE RECORD JUST READ.
!*---------------------------------------------------------------------
MOVE ARCHIVE-EMPLOYEE-RECORD TO EMPLOYEE.
MOVE EMP-ID-0415 TO WORK-ARC-ID.
CONTINUE.
Dialog RESD3

Dialog RESD3 restores coverage records associated with the restored employee record. If an employee record is read, control passes back automatically to dialog RESD2.

The dialog definition for RESD2 is illustrated below:

The mapin operation and response process are shown below.

Dialog RESD3: Mapin Operation

<table>
<thead>
<tr>
<th>External field</th>
<th>Internal field</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE-TYPE</td>
<td>$RESPONSE</td>
</tr>
<tr>
<td>ARCHIVE-COVERAGE-RECORD</td>
<td>ARCHIVE-COVERAGE-RECORD</td>
</tr>
</tbody>
</table>

Dialog RESD3: Response Process

*RESPD3-RESPONSE

*
Executing the Application

You execute the application by executing the batch program ADSBATCH, as described in Runtime Considerations (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

You should be able to use the same JCL and control statements that you used in the archive application (presented earlier in this section), with the exception of the ENTRY POINT control statement, which is shown below for the restore application:

ENTRY POINT TASK RESAPPL.
The contents of INFILE1 before the application is executed, and the contents of REPFILE1, SUSFILE1, and ADSLOGA after the application is executed are shown below.

Note: The restore application assumes that both the input file of employee ids and the archive file are ordered by employee id. Make sure that your input file is ordered properly.

INFILE1

3000  
3010  
4000  
5001  

REPFILE1

<table>
<thead>
<tr>
<th>EMPLOYEE ID</th>
<th>NAME</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>STERNS</td>
<td>JOSEPH</td>
</tr>
<tr>
<td>3010</td>
<td>PETERSON</td>
<td>RUTH</td>
</tr>
<tr>
<td>4000</td>
<td></td>
<td>EMPLOYEE NOT FOUND</td>
</tr>
<tr>
<td>5001</td>
<td>WATSON</td>
<td>BRIAN</td>
</tr>
</tbody>
</table>

SUSFILE1

3010  
4000  

ADSLOGA

IDMS DC506801 V1 SUSPENSE FILE SUSFILE1 RECORD# 1 IMAGE IS '3010'  
IDMS DC506801 V1 SUSPENSE FILE SUSFILE1 RECORD# 2 IMAGE IS '4000'  
***EOF ON INPUT***  
IDMS DC507001 V1 BASE FILE INFILE1 CLOSED.  
IDMS DC507003 V1 4 LOGICAL RECORDS READ.  
IDMS DC507002 V1 BASE DATASET MEN.C0600.MPK.INFILE1 CLOSED.  
IDMS DC507003 V1 1 PHYSICAL RECORDS READ.  
IDMS DC507001 V1 BASE FILE SUSFILE1 CLOSED.  
IDMS DC507004 V1 2 LOGICAL RECORDS WRITTEN.  
IDMS DC507002 V1 BASE DATASET MEN.C0600.MPK.SUSFILE1 CLOSED.  
IDMS DC507004 V1 1 PHYSICAL RECORDS WRITTEN.  
IDMS DC507001 V1 SUPPLEMENTARY FILE ARCFILE CLOSED.  
IDMS DC507003 V1 3 LOGICAL RECORDS READ.  
IDMS DC507001 V1 BASE FILE ARCFILE CLOSED.  
IDMS DC507003 V1 3 LOGICAL RECORDS READ.  
IDMS DC507002 V1 BASE DATASET MEN.C0600.MPK.ARCFILE CLOSED.  
IDMS DC507003 V1 1 PHYSICAL RECORDS READ.  
IDMS DC507001 V1 BASE FILE REPFILE1 CLOSED.  
IDMS DC507004 V1 6 LOGICAL RECORDS WRITTEN.  
IDMS DC507002 V1 BASE DATASET MEN.C0600.MPK.REPFILE1 CLOSED.  
IDMS DC507004 V1 1 PHYSICAL RECORDS WRITTEN.

Employee-Record Report Application

The employee-record report application produces a report of employees by department. Sample output is shown below:

Sample Output
Employee listing by department

<table>
<thead>
<tr>
<th>DEPARTMENT ID</th>
<th>DEPARTMENT</th>
<th>TOTAL EMPLOYEES</th>
</tr>
</thead>
<tbody>
<tr>
<td>5200</td>
<td>THERMOREGULATION</td>
<td>6</td>
</tr>
<tr>
<td>3100</td>
<td>INTERNAL SOFTWARE</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>TOTAL EMPLOYEES ALL DEPARTMENTS</td>
<td>61</td>
</tr>
</tbody>
</table>

**Employee-record Flow Example**

The diagram below shows the file access, components, and flow of control for the application.
Files Accessed

This application accesses the following files.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPFILE1</td>
<td>Output file</td>
<td>Contains the report</td>
</tr>
<tr>
<td>ADSLOGA</td>
<td>Log file</td>
<td>Contains informational and error messages produced by the application</td>
</tr>
</tbody>
</table>

Dialogs

The application consists of the following dialogs.

<table>
<thead>
<tr>
<th>Name Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPT</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>EDIT</td>
</tr>
<tr>
<td>DEPT</td>
</tr>
<tr>
<td>HDR</td>
</tr>
<tr>
<td>COL</td>
</tr>
<tr>
<td>MHD</td>
</tr>
<tr>
<td>R</td>
</tr>
</tbody>
</table>
**Name Description**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLINE OUT</td>
<td>Writes detail lines, as shown below:</td>
</tr>
<tr>
<td>HDRLI NE</td>
<td>Writes page header lines, as shown below:</td>
</tr>
</tbody>
</table>

**Steps**

To create the application, you perform the following steps:

1. Describe the layouts of the records used in the application.
2. Define the file maps that associate file records with variable storage.
3. Define the process modules for the application’s dialogs.
4. Define the dialogs.

Note that you do not have to describe the report file in the data dictionary; you can use file entity IDD-REPFIL1, which you already described as part of the archive application.

The steps you perform to define the report application are described below, followed by a discussion of executing the application.

**Step 1 Describe the Records in the Data Dictionary**

All records (except subschema records) used in the application must be described in the data dictionary. The employee-record report application uses the following records.

**Records Used**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK-PRINT-EDIT</td>
<td>Contains miscellaneous variable fields required by the application</td>
</tr>
<tr>
<td>EMPL-DETAIL</td>
<td>Describes the layout of detail and detail header lines in the report file</td>
</tr>
<tr>
<td>REPT-HEADER</td>
<td>Describes the layout of page header, department header, and total lines in the report file</td>
</tr>
</tbody>
</table>

You can define the records by using the IDD DDDL compiler or the IDD menu facility. The record definitions are illustrated below.

**WORK-PRINT-EDIT**

```
WORK-PRINT-EDIT.
  03 WK-HDG   PICTURE IS X(100).
  03 WK-LENGTH PICTURE IS 999 USAGE IS COMP.
  03 WK-START-POS PICTURE IS 999 USAGE IS COMP.
  03 WK-LINE-NBR PICTURE IS 99 USAGE IS COMP.
  03 WK-CURR-DATE PICTURE IS X(8).
  03 WK-DATE-HOLD PICTURE IS 9(6).
```
Step 2 Define the File Maps

You define a file map for each file record layout used in the application. The file maps in this application are described below.

File Maps
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDRMAP</td>
<td>Associates record REPT-HEADER with variable storage</td>
</tr>
<tr>
<td>DTLMAP</td>
<td>Associates record EMPL-DETAIL with variable storage</td>
</tr>
</tbody>
</table>

You define the maps by using the CA IDMS/DC mapping facility. The file map definitions are illustrated below.

**HDRMAP**

Internal records: None

External file: IDD-REPFILE1

External record: REPT-HEADER

<table>
<thead>
<tr>
<th>External field</th>
<th>Internal field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL-CHAR</td>
<td>CTRL-CHAR</td>
</tr>
<tr>
<td>LEFT</td>
<td>LEFT</td>
</tr>
<tr>
<td>CENTER</td>
<td>CENTER</td>
</tr>
<tr>
<td>RIGHT</td>
<td>RIGHT</td>
</tr>
</tbody>
</table>

**DTLMAP**

Internal records: None

External file: IDD-REPFILE1

External record: EMPL-DETAIL

<table>
<thead>
<tr>
<th>External field</th>
<th>Internal field</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTL-CTRL-CHAR</td>
<td>DTL-CTRL-CHAR</td>
</tr>
<tr>
<td>DTL-EMPL</td>
<td>DTL-EMPL</td>
</tr>
<tr>
<td>DTL-NAME</td>
<td>DTL-NAME</td>
</tr>
<tr>
<td>DTL-STATUS</td>
<td>DTL-STATUS</td>
</tr>
<tr>
<td>DTL-SS-NBR</td>
<td>DTL-SS-NBR</td>
</tr>
<tr>
<td>DTL-SS-TITLE</td>
<td>DTL-SS-TITLE</td>
</tr>
<tr>
<td>DTL-START</td>
<td>DTL-START</td>
</tr>
<tr>
<td>DTL-TERM</td>
<td>DTL-TERM</td>
</tr>
</tbody>
</table>

**Step 3 Define the Process Modules**

As the next step in creating the application, you define process modules consisting of Application Development System process commands. You can define process modules by using the IDD DDDL compiler or the IDD menu facility.

The process modules required by the application are presented under "Step 4: Define the Dialogs," together with the discussion of the dialogs with which they are associated. In this way, you can see how the modules fit into the application structure.

⚠️ **Note:** The process modules shown in Step 4 have embedded pointers, indicated by an exclamation point (!) that precedes the comments. You do not have to key in these comments.
Step 4 Define the Dialogs

The next step in defining the application is to define its dialogs. A dialog is a collection of application components created in earlier steps, including file maps and process modules. You can define a dialog by using the online dialog compiler.

The dialogs in the report application are described below.

Dialogs

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPT</td>
<td>Acts as a mainline routine, reading database records, setting up report lines, and passing control to other dialogs, as required by the application; writes department and report total lines to the report file</td>
</tr>
<tr>
<td>EDIT</td>
<td></td>
</tr>
<tr>
<td>DEPT</td>
<td>Writes department header lines to the report file</td>
</tr>
<tr>
<td>HDR</td>
<td></td>
</tr>
<tr>
<td>COL</td>
<td>Writes detail column header lines to the report file</td>
</tr>
<tr>
<td>MHD</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>DLINE</td>
<td>Writes detail lines to the report file</td>
</tr>
<tr>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>HDRLI</td>
<td>Writes page header lines to the report file</td>
</tr>
<tr>
<td>NE</td>
<td></td>
</tr>
</tbody>
</table>

The definitions for these dialogs are provided separately below, along with the process modules associated with the dialogs and the mapout operations these dialogs perform at runtime. For an illustration of how these dialogs fit together, see the diagram earlier in this section.

Dialog REPTEDIT

Dialog REPTEDIT executes at the beginning of the application. REPTEDIT reads database records required for the report, sets up report lines, and passes control to other dialogs, as required by the application. REPTEDIT also writes department and report total lines to the report file.

The dialog definition for REPTEDIT is illustrated below:
The premap process and mapout operation are shown below.

**Dialog DEPTEDIT: Premap Process**

```
!**********************************************************************
!*EMPL-REPT-EDIT                                                     *
!**********************************************************************
IF FIRST-TIME
DO.
  MOVE DATE TO WK-DATE-HOLD.  /*SET UP CURRENT DATE
  CALL CVRTDATE.
  MOVE WK-DATE-HOLD TO WK-CURR-DATE.
  CALL PAGHDR.  /*SET UP FIRST PAGE.
  OBTAIN FIRST DEPARTMENT WITHIN ORG-DEMO-REGION.
END.
ELSE
  OBTAIN NEXT DEPARTMENT WITHIN ORG-DEMO-REGION.
  IF DB-ANY-ERROR
    DO.
    !
    /*SET UP GRAND TOTAL EMPLOYEE COUNT LINE TO LOOK*  
  AS FOLLOWS:  /*
  !  TOTAL EMPLOYEES ALL DEPARTMENTS: ZZZ9  *
```
MOVE SPACES TO CENTER.
MOVE INSERT(CENTER,'TOTAL EMPLOYEES ALL DEPARTMENTS: ',68)
TO CENTER.
MOVE WK-TOT-EE-COUNT TO WK-DEPT-EE-COUNT.
CALL TOTEDIT.
WRITE TO LOG MSG TEXT 'END OF EMPLOYEE REPORT RUN'.
WRITE TRANSACTION RETURN. !*LAST TOTAL LINE AND LEAVE APPL END.
!
MOVE DEPT-ID-0410 TO WK-DEPT-PRINT. !*SET UP CURRENT DEPARTMENT ID
IF WK-LINE-NBR GE 52 !*CHECK FOR PAGE BREAK ...
CALL PAGHDR. !*... AND START NEW PAGE IF NEEDED
LINK NOSAVE TO 'DEPTHDR'. !*PUT OUT DEPARTMENT HEADER
LINK NOSAVE TO 'COLMHDR'. !*PUT OUT COLUMN HEADER
MOVE '0' TO DTL-CTRL-CHAR. !*DOUBLE SPACE NEXT DETAIL
OBTAIN FIRST EMPLOYEE WITHIN DEPT-EMPLOYEE.
!
WHILE NOT DB-END-OF-SET
REPEAT.
MOVE EMP-ID-0415 TO DTL-EMP-ID. !*EDIT ONE EMPLOYEE ... DETAIL LINE
MOVE EMP-LAST-NAME-0415 TO DTL-LAST-NAME.
MOVE EMP-FIRST-NAME-0415 TO DTL-FIRST-NAME.
MOVE 'UNK' TO DTL-STAT-CODE.
IF ACTIVE-0415
MOVE 'ACT' TO DTL-STAT-CODE.
IF ST-DISABIL-0415
MOVE 'STD' TO DTL-STAT-CODE.
IF LT-DISABIL-0415
MOVE 'LDI' TO DTL-STAT-CODE.
IF LEAVE-OF-ABSENCE-0415
MOVE 'LOA' TO DTL-STAT-CODE.
IF TERMINATED-0415
MOVE 'TRM' TO DTL-STAT-CODE.
MOVE SS-NUMBER-0415 TO DTL-SS-NBR.
MOVE START-DATE-0415 TO WK-DATE-HOLD.
CALL CVRTDATE.
MAKE WK-DATE-WORK TO DTL-START-DATE.
MOVE TERMINATION-DATE-0415 TO WK-DATE-HOLD.
CALL CVRTDATE.
MOVE WK-DATE-WORK TO DTL-TERM-DATE.
IF WK-LINE-NBR GE 58 !*PAGE BREAK?
DO.
CALL PAGHDR. !*PUT OUT PAGE HEADERS
LINK NOSAVE TO 'COLMHDR'. !*AND COLUMN HEADER
MOVE '0' TO DTL-CTRL-CHAR. !*DOUBLE SPACE NEXT DETAIL
END.
LINK NOSAVE TO 'DLINEOUT'. !*PUT OUT A DETAIL LINE
ADD 1 TO WK-DEPT-EE-COUNT. !*PLUS 1 TO EMPLOYEE COUNT
MOVE SPACE TO DTL-CTRL-CHAR. !*SINGLE SPACE NEXT DETAIL
OBTAIN NEXT EMPLOYEE WITHIN DEPT-EMPLOYEE.
END.
!
!*************************************************
!*-SET UP DEPARTMENT TOTAL EMPLOYEE COUNT LINE TO*
!* LOOK AS FOLLOWS:               *
!* DEPARTMENT XXXX TOTAL EMPLOYEES: ZZZ9 *
!*************************************************
MOVE CONCATENATE('DEPARTMENT ',WK-DEPT-PRINT,' TOTAL EMPLOYEES:')
TO WK-HDG.
MOVE SPACES TO CENTER.
MOVE INSERT(CENTER,SUBSTRING(WK-HDG,1,32),68) TO CENTER.
CALL TOTEDIT.
ADD WK-DEPT-EE-COUNT TO WK-TOT-EE-COUNT.!*ROLL DEPT INTO GRAND TOTAL
MOVE 0 TO WK-DEPT-EE-COUNT. !*INIT DEPT TOTAL
ADD 2 TO WK-LINE-NBR. !*INCREASE LINE COUNT
WRITE TRANSACTION CONTINUE. !*OUTPUT DEPT TOTAL LINE
!*SUBROUTINE TO EDIT AN EMPLOYEE COUNT TOTAL LINE*
!*-----------------------------------------------------------------------*
DEFINE TOTEDIT.

MOVE LEFT-LEFT-JUSTIFY(WK-DEPT-EE-COUNT) TO RIGHT.
MOVE SPACES TO LEFT.
MOVE '0' TO CTRL-CHAR.  !*DOUBLE SPACE IT
GOBACK.

!*-----------------------------------------------------------------------*
!*SUBROUTINE TO PUT OUT PAGE HEADER LINES                                *
!*-----------------------------------------------------------------------*
DEFINE PAGHDR.

MOVE 0 TO WK-LINE-NBR.    !*RESET LINE COUNTER
ADD 1 TO WK-PAGE-NBR.    !*INCREASE PAGE NUMBER
MOVE WK-PAGE-NBR TO WK-PAGE-PRINT.  !*SET IT UP TO PRINT
MOVE '1' TO CTRL-CHAR.    !*NEW PAGE
MOVE 'FCI0001' TO LEFT.  !*REPORT ID
MOVE 'FICTIONAL COMPANY, INC.' !*COMPANY NAME
TO WK-HDG.
CALL CENTJUST.  !*GO CENTER IT
MOVE 'DATE PAGE' TO RIGHT.  !*DATE/PAGE HEADERS
LINK NOSAVE TO 'HDRLINE'.    !*OUTPUT 1ST HEADER LINE
MOVE SPACES TO CTRL-CHAR.    !*SINGLE SPACE
MOVE SPACES TO LEFT.
MOVE 'EMPLOYEE LISTING BY DEPARTMENT' !*REPORT TITLE
TO WK-HDG.
CALL CENTJUST.  !*GO CENTER IT
MOVE CONCATENATE(' ',WK-CURR-DATE,' ',WK-PAGE-PRINT) TO RIGHT.
LINK NOSAVE TO 'HDRLINE'.
GOBACK.

!*-----------------------------------------------------------------------*
!*SUBROUTINE TO CENTER JUSTIFY HEADER TITLES                              *
!*-----------------------------------------------------------------------*
DEFINE CENTJUST.

MOVE STRING-LENGTH(EXTRACE(WK-HDG)) TO WK-LENGTH.
COMPUTE WK-START-POS TRUNCATED = ((100 - WK-LENGTH) / 2 + 1.
MOVE SPACES TO CENTER.
MOVE INSERT(CENTER,EXTRACT(WK-HDG),WK-START-POS) TO CENTER.
GOBACK.

!*-----------------------------------------------------------------------*
!*SUBROUTINE TO CONVERT A DATE FROM YYMMDD TO MM/DD/YY                    *
!*-----------------------------------------------------------------------*
DEFINE CVRTDATE.

MOVE SUBSTRING(WK-DATE-X,3,2) TO WK-MO.
MOVE SUBSTRING(WK-DATE-X,5,2) TO WK-DAY.
MOVE SUBSTRING(WK-DATE-X,1,2) TO WK-YR.
GOBACK.

Dialog REPTEDIT: Mapout Operation

Internal field     External field
CTRL-CHAR.................>CTRL-CHAR
LEFT......................>LEFT
CENTER....................>CENTER
RIGHT.....................>RIGHT
Dialog DEPTHDR

Dialog DEPTHDR executes when it receives control from REPTEDIT. DEPTHDR writes department header lines to the report file.

The dialog definition for DEPTHDR is illustrated below:

![Diagram of DEPTHDR](image)

Premap process

DEPT-HDR-EDIT

Mapout

Map name: HDRMAP
Ddname: EMPLRPT

Work records

WORK-PRINT-EDIT
DEPARTMENT

Subschema

EMPSS01

IDMSDB--Dialog DEPTHDR

The premap process and mapout operation are shown below.

**Dialog DEPTHDR: Premap Process**

```plaintext
!*DEPT-HDR-EDIT

MOVE '0' TO CTRL-CHAR.
MOVE 'DEPARTMENT ID:' TO LEFT.
MOVE CONCATENATE(DEPT-ID-0410,' ',DEPT-NAME-0410) TO CENTER.
ADD 2 TO WK-LINE-NBR.
WRITE TRANSACTION RETURN.
```

13-Jan-2018
Dialog DEPTHDR: Mapout Operation

Internal field          External field
CTRL-CHAR.......................>CTRL-CHAR
LEFT...........................>LEFT
CENTER.........................>CENTER
RIGHT.........................>RIGHT

Dialog COLMHDR

Dialog COLMHDR executes when it receives control from REPTEDIT. COLMHDR writes detail column header lines to the report file.

The dialog definition for COLMHDR is illustrated below:

![Diagram of COLMHDR dialog]

Note: EMPL-DETAIL should be associated with the dialog as a new copy record.
The premap process and mapout operation are shown below.

**Dialog COLMHDR: Premap Process**

```
!**********************************************************************
!*COLM-HDR-EDIT                                                       *
!**********************************************************************
MOVE '0' TO DTL-CTRL-CHAR.
MOVE 'EMP ID' TO DTL-EMPL.
MOVE SPACES TO DTL-NAME.
MOVE INSERT(DTL-NAME, '---NAME---',2) TO DTL-NAME.
MOVE 'STATUS' TO DTL-STATUS.
MOVE 'SOC SEC ' TO DTL-SS-TITLE.
MOVE 'START DATE' TO DTL-START.
MOVE 'TERM DATE' TO DTL-TERM.
ADD 2 TO WK-LINE-NBR.
WRITE TRANSACTION RETURN.
```

**Dialog COLMHDR: Mapout Operation**

<table>
<thead>
<tr>
<th>Internal field</th>
<th>External field</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTL-CTRL-CHAR</td>
<td>DTL-CTRL-CHAR</td>
</tr>
<tr>
<td>DTL-EMPL</td>
<td>DTL-EMPL</td>
</tr>
<tr>
<td>DTL-NAME</td>
<td>DTL-NAME</td>
</tr>
<tr>
<td>DTL-STATUS</td>
<td>DTL-STATUS</td>
</tr>
<tr>
<td>DTL-SS-NBR</td>
<td>DTL-SS-NBR</td>
</tr>
<tr>
<td>DTL-SS-TITLE</td>
<td>DTL-SS-TITLE</td>
</tr>
<tr>
<td>DTL-START</td>
<td>DTL-START</td>
</tr>
<tr>
<td>DTL-TERM</td>
<td>DTL-TERM</td>
</tr>
</tbody>
</table>

**Dialog DLINEOUT**

Dialog DLINEOUT executes when it receives control from REPTEDIT. DLINEOUT writes detail lines to the report file.

The dialog definition for DLINEOUT is illustrated below:
The premap process and mapout operation are shown below.

**Dialog DLINEOUT: Premap Process**

```plaintext
!*DETAIL-WRITE
!*DETAIL-WRITE

IF DTL-CTRL-CHAR = '0'
    ADD 2 TO WK-LINE-NBR.
ELSE
    ADD 1 TO WK-LINE-NBR.
WRITE TRANSACTION RETURN.
```

**Dialog DLINEOUT: Mapout Operation**

<table>
<thead>
<tr>
<th>Internal field</th>
<th>External field</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTL-CTRL-CHAR</td>
<td>DTL-CTRL-CHAR</td>
</tr>
<tr>
<td>DTL-EMPL</td>
<td>DTL-EMPL</td>
</tr>
<tr>
<td>DTL-NAME</td>
<td>DTL-NAME</td>
</tr>
<tr>
<td>DTL-STATUS</td>
<td>DTL-STATUS</td>
</tr>
<tr>
<td>DTL-SS-NBR</td>
<td>DTL-SS-NBR</td>
</tr>
<tr>
<td>DTL-SS-TITLE</td>
<td>DTL-SS-TITLE</td>
</tr>
</tbody>
</table>
Dialog HDRLINE

Dialog HDRLINE executes when it receives control from REPTEDIT. HDRLINE writes page headers to the report file.

The dialog definition for HDRLINE is illustrated below:

```
<table>
<thead>
<tr>
<th>Premap process</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEADER-WRITE</td>
</tr>
<tr>
<td>Mapout</td>
</tr>
<tr>
<td>Map name: DTLMAP</td>
</tr>
<tr>
<td>Ddname: EMPLRPT</td>
</tr>
<tr>
<td>Work records</td>
</tr>
<tr>
<td>WORK-PRINT-EDIT</td>
</tr>
<tr>
<td>Subschema</td>
</tr>
<tr>
<td>EMPSS01</td>
</tr>
</tbody>
</table>
```

IDMSDB--Dialog HDRLINE

The premap process and mapout operation are shown below.

**Dialog HDRLINE: Premap Process**

```
!*---------------------------------------------------------------------
!*HEADER-WRITE
!*---------------------------------------------------------------------
ADD 1 TO WK-LINE-NBR.
WRITE TRANSACTION RETURN.
```

**Dialog HDRLINE: Mapout Operation**
Executing the Application

You execute the application by executing the batch program ADSBATCH, as described in Runtime Considerations (https://docops.ca.com/display/IDMS19/Runtime+Considerations).

You should be able to use the same JCL and control statements for the report and log files that you used in the previous applications. The only exception is the ENTRY POINT control statement, which is shown below for the report application:

ENTRY POINT DIALOG REPTEDIT.

Report File Listing

A partial listing of the report file after the application is executed is shown below:

<table>
<thead>
<tr>
<th>DEPARTMENT ID: 5200</th>
<th>THERMOREGULATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP ID</td>
<td>NAME--</td>
</tr>
<tr>
<td>0479</td>
<td>CLOTH</td>
</tr>
<tr>
<td>0329</td>
<td>FINN</td>
</tr>
<tr>
<td>0469</td>
<td>KASPAR</td>
</tr>
<tr>
<td>4014</td>
<td>SHEPARD</td>
</tr>
<tr>
<td>0355</td>
<td>TIME</td>
</tr>
<tr>
<td>0439</td>
<td>WILCO</td>
</tr>
</tbody>
</table>

DEPARTMENT 5200 TOTAL EMPLOYEES: 6

<table>
<thead>
<tr>
<th>DEPARTMENT ID: 3100</th>
<th>INTERNAL SOFTWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP ID</td>
<td>NAME</td>
</tr>
<tr>
<td>0024</td>
<td>DOUGH</td>
</tr>
<tr>
<td>0029</td>
<td>GALLWAY</td>
</tr>
<tr>
<td>0003</td>
<td>GARFIELD</td>
</tr>
<tr>
<td>0028</td>
<td>GRANGER</td>
</tr>
<tr>
<td>0027</td>
<td>HEAROWITZ</td>
</tr>
<tr>
<td>0020</td>
<td>JACOBI</td>
</tr>
<tr>
<td>0019</td>
<td>JENSEN</td>
</tr>
<tr>
<td>0035</td>
<td>LITERATA</td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td>0023</td>
<td>O'HEARN</td>
</tr>
<tr>
<td>0021</td>
<td>TYRO</td>
</tr>
</tbody>
</table>

DEPARTMENT 3100 TOTAL EMPLOYEES: 10
TOTAL EMPLOYEES ALL DEPARTMENTS: 61