CA IDMS - 19.0
Using CA ADS

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

**Using CA ADS Alive** ................................................................. 14

Overview .................................................................................. 14

CA ADS Alive A Powerful Interactive Tool ................................ 14

Using CA ADS Alive to Debug and Test Dialogs ...................... 14

Dialog Animation ........................................................................ 15
  Dialog Animation Capabilities ............................................. 15
  Two Animation Modes ......................................................... 15
  Specifying and Altering Stop Locations .................................. 16
  Post-Abort Browse Facility .................................................... 16
  Displaying & Modifying Record/Element Contents ............... 16

Suggestions for Testing ............................................................... 16

CA ADS Alive Session ................................................................ 17
  Creating a CA ADS Alive Animation Session ....................... 17
    Create An Animation Session .......................................... 17
    Animate the Dialog ......................................................... 19
  Typical CA ADS Alive Screen .............................................. 20
    Transfer Facility ............................................................ 21
  Post-abort Browse .............................................................. 22
  Record/Element Display & Modification .............................. 23
    Displaying a Record using the RECORD Command ........... 23
    Displaying a Record using the ELEMENT Command ........ 24

CA ADS Alive Operations .......................................................... 25
  Online Documentation Print Utility ..................................... 26
  CA ADS Alive Customization Macros ................................. 27
  Reviewing the Post-Abort Browse Queue ............................ 27
    The Queue Review Procedure ........................................ 28
    The Queue Review Screen .............................................. 28

**Using ADS Batch** .................................................................. 29

Introduction to ADS Batch ......................................................... 31
  Example 1 Employee-record Archiving Application ............... 31
  Example 2 Employee-record Restore Application ................ 32

CA ADS Batch Concepts ............................................................ 33
  Input and Output Files and File Maps .................................. 34
  Suspense Files ..................................................................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Files</td>
<td>38</td>
</tr>
<tr>
<td>Log File Information</td>
<td>39</td>
</tr>
<tr>
<td>Log File Prefix</td>
<td>39</td>
</tr>
<tr>
<td>Archiving</td>
<td>40</td>
</tr>
<tr>
<td>Batch Dialog Structure</td>
<td>40</td>
</tr>
<tr>
<td>CA ADS Batch Dialog Components</td>
<td>41</td>
</tr>
<tr>
<td>Batch Control Events</td>
<td>41</td>
</tr>
<tr>
<td>Batch Response Field Values</td>
<td>42</td>
</tr>
<tr>
<td>Batch Dialog Execution</td>
<td>42</td>
</tr>
<tr>
<td>Application Structure</td>
<td>46</td>
</tr>
<tr>
<td>Disallowed Functions</td>
<td>46</td>
</tr>
<tr>
<td>Application Flow of Control</td>
<td>47</td>
</tr>
<tr>
<td>Accessing Input Files with Multiple Record Layouts</td>
<td>48</td>
</tr>
<tr>
<td>Application Compiler</td>
<td>51</td>
</tr>
<tr>
<td>General Options Screens</td>
<td>51</td>
</tr>
<tr>
<td>Global Records Screen</td>
<td>53</td>
</tr>
<tr>
<td>Task Codes Screen</td>
<td>53</td>
</tr>
<tr>
<td>Response/Function List Screen</td>
<td>54</td>
</tr>
<tr>
<td>Response Definition Screen</td>
<td>55</td>
</tr>
<tr>
<td>Function Definition Screen</td>
<td>56</td>
</tr>
<tr>
<td>Dialog Compiler</td>
<td>57</td>
</tr>
<tr>
<td>Dialog Compiler in Online Mode</td>
<td>58</td>
</tr>
<tr>
<td>Main Menu Screen</td>
<td>58</td>
</tr>
<tr>
<td>Map Specifications Screen</td>
<td>59</td>
</tr>
<tr>
<td>Database Specifications</td>
<td>60</td>
</tr>
<tr>
<td>Options and Directives Screen</td>
<td>60</td>
</tr>
<tr>
<td>Records and Tables Screen</td>
<td>61</td>
</tr>
<tr>
<td>Process Modules Screen</td>
<td>62</td>
</tr>
<tr>
<td>Dialog Compiler in Batch Mode (ADSOBCOM)</td>
<td>63</td>
</tr>
<tr>
<td>CA IDMS/DC Mapping Facility</td>
<td>63</td>
</tr>
<tr>
<td>Batch Automatic Editing and Error Handling</td>
<td>63</td>
</tr>
<tr>
<td>Variable Array Processing</td>
<td>64</td>
</tr>
<tr>
<td>Online Mapping Facility</td>
<td>65</td>
</tr>
<tr>
<td>Map Definition Screen</td>
<td>66</td>
</tr>
<tr>
<td>Extended Map Definition Screen</td>
<td>66</td>
</tr>
<tr>
<td>File Field Selection Screen</td>
<td>66</td>
</tr>
<tr>
<td>File Field Edit Screen</td>
<td>68</td>
</tr>
<tr>
<td>Batch Map Compiler (RHDCMAP1)</td>
<td>70</td>
</tr>
<tr>
<td>ADSOBSYS</td>
<td>70</td>
</tr>
<tr>
<td>z/VSE File Characteristics Program</td>
<td>75</td>
</tr>
<tr>
<td>DEFINE CHARACTERISTICS Statement</td>
<td>77</td>
</tr>
<tr>
<td>ACCEPT CHARACTERISTICS statement</td>
<td>80</td>
</tr>
</tbody>
</table>
### CA ADS Batch Sample Applications

- **Employee-record Archive Application**
  - Step 1 Describe the Files in the Data Dictionary ................................................................. 91
  - Step 2 Describe the Records in the Data Dictionary ............................................................. 91
  - Step 3 Define the File Maps .................................................................................................. 93
  - Step 4 Define the Process Modules ....................................................................................... 94
  - Step 5 Define the Dialogs ...................................................................................................... 94
  - Executing the Application .................................................................................................... 105

- **Employee-Record Restore Application**
  - Step 1 Define the Application Structure .............................................................................. 108
  - Step 2 Define the Process Modules ....................................................................................... 109
  - Step 3 Define the Dialogs ...................................................................................................... 110
  - Executing the Application .................................................................................................... 119

- **Employee-Record Report Application**
  - Step 1 Describe the Records in the Data Dictionary ............................................................. 123
  - Step 2 Define the File Maps .................................................................................................. 124
  - Step 3 Define the Process Modules ....................................................................................... 125
  - Step 4 Define the Dialogs ...................................................................................................... 126
  - Executing the Application .................................................................................................... 135

### Using ADS for IDMS

- **Building a Prototype** ......................................................................................................... 136
  - Stage 1 Building the Basic Prototype ..................................................................................... 137
    - Compiling the Application (ADSA) ....................................................................................... 138
    - Compiling the Maps (MAPC) .............................................................................................. 138
    - Compiling the Dialogs (ADSC) ........................................................................................... 139
    - User Review .......................................................................................................................... 140
  - Stage 2 Adding Process Logic and Data Retrieval ................................................................. 140
    - ADSA Enhancements ............................................................................................................ 140
    - Populating the Data Dictionary (IDD) .................................................................................. 141
    - MAPC Enhancements ........................................................................................................... 141
    - ADSC Enhancements ............................................................................................................ 141
  - Stage 3 Refining the Maps and Processes .............................................................................. 142
- **Designing Maps** .................................................................................................................. 142
- **Design Standards for a Dialog Map** .................................................................................... 143

---

**Using CA ADS 6**
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping Procedures</td>
<td>143</td>
</tr>
<tr>
<td>Choosing Menu Map</td>
<td>144</td>
</tr>
<tr>
<td>System-Defined Menu Maps</td>
<td>144</td>
</tr>
<tr>
<td>User-Defined Menu Maps</td>
<td>145</td>
</tr>
<tr>
<td>Reformating and Recompiling the System-Defined Menu</td>
<td>145</td>
</tr>
<tr>
<td>Designing a Menu/Dialog</td>
<td>146</td>
</tr>
<tr>
<td>Designing Dialog Maps</td>
<td>147</td>
</tr>
<tr>
<td>Designing Dialog</td>
<td>147</td>
</tr>
<tr>
<td>Dialog Characteristics</td>
<td>147</td>
</tr>
<tr>
<td>Dialog Level</td>
<td>148</td>
</tr>
<tr>
<td>Dialog Status</td>
<td>148</td>
</tr>
<tr>
<td>Dialog Control</td>
<td>150</td>
</tr>
<tr>
<td>ADS - Design Considerations</td>
<td>150</td>
</tr>
<tr>
<td>Database Currencies</td>
<td>150</td>
</tr>
<tr>
<td>Database, Work, and Map Records</td>
<td>151</td>
</tr>
<tr>
<td>Logical Records</td>
<td>154</td>
</tr>
<tr>
<td>NEW COPY Records</td>
<td>154</td>
</tr>
<tr>
<td>Working Storage Areas</td>
<td>154</td>
</tr>
<tr>
<td>Queue Records</td>
<td>155</td>
</tr>
<tr>
<td>Scratch Records</td>
<td>155</td>
</tr>
<tr>
<td>Extended Run Units</td>
<td>156</td>
</tr>
<tr>
<td>Longterm Locks</td>
<td>157</td>
</tr>
<tr>
<td>Global Records</td>
<td>158</td>
</tr>
<tr>
<td>ADS Naming Conventions</td>
<td>162</td>
</tr>
<tr>
<td>Naming Application Entities</td>
<td>162</td>
</tr>
<tr>
<td>Naming Database Information</td>
<td>164</td>
</tr>
<tr>
<td>Performance Considerations 1</td>
<td>166</td>
</tr>
<tr>
<td>CA ADS System Generation</td>
<td>166</td>
</tr>
<tr>
<td>Statements</td>
<td>166</td>
</tr>
<tr>
<td>Allocating Primary and Secondary Storage Pools</td>
<td>168</td>
</tr>
<tr>
<td>Relocating Resources</td>
<td>169</td>
</tr>
<tr>
<td>Specifying the Number of Online Tasks and External Request Units</td>
<td>170</td>
</tr>
<tr>
<td>ADS Resource Management</td>
<td>171</td>
</tr>
<tr>
<td>Monitoring Tools</td>
<td>172</td>
</tr>
<tr>
<td>Task Processing Support</td>
<td>172</td>
</tr>
<tr>
<td>Variable Storage Pool</td>
<td>173</td>
</tr>
<tr>
<td>Program Pool Storage</td>
<td>174</td>
</tr>
<tr>
<td>Database Locks</td>
<td>174</td>
</tr>
<tr>
<td>Disk I/O</td>
<td>175</td>
</tr>
<tr>
<td>Terminal I/O</td>
<td>175</td>
</tr>
<tr>
<td>CPU Usage</td>
<td>176</td>
</tr>
<tr>
<td>Conserving Resources</td>
<td>176</td>
</tr>
<tr>
<td>Step 3 Exit from the Application</td>
<td>..........................................................</td>
</tr>
<tr>
<td>Summary 2</td>
<td>...............................................................................</td>
</tr>
<tr>
<td>Modifying the Application Structure Using ADSA</td>
<td>.................................................</td>
</tr>
<tr>
<td>Steps to Modify Department Application</td>
<td>........................................................</td>
</tr>
<tr>
<td>Step 1 Retrieve the Application to be Modified</td>
<td>.............................................</td>
</tr>
<tr>
<td>Step 2 Select Responses and Functions</td>
<td>..................................................</td>
</tr>
<tr>
<td>Step 3 Modify the EXIT Response</td>
<td>....................................................</td>
</tr>
<tr>
<td>Step 4 Modify the ADDDEP Function</td>
<td>..................................................</td>
</tr>
<tr>
<td>Step 5 Recompile the Application</td>
<td>....................................................</td>
</tr>
<tr>
<td>Exit From ADSA 1</td>
<td>.......................................................................</td>
</tr>
<tr>
<td>Execute the Application</td>
<td>..................................................................</td>
</tr>
<tr>
<td>Summary 3</td>
<td>...............................................................................</td>
</tr>
<tr>
<td>Modifying a Map Using MAPC</td>
<td>..................................................................</td>
</tr>
<tr>
<td>Modifying a Map Using MAPC 1</td>
<td>..................................................................</td>
</tr>
<tr>
<td>Step 1 Retrieve the Map To Be Modified</td>
<td>..................................................</td>
</tr>
<tr>
<td>Step 2 Add and Select Map Fields</td>
<td>.....................................................</td>
</tr>
<tr>
<td>Step 3 Edit the Selected Fields</td>
<td>........................................................</td>
</tr>
<tr>
<td>Step 4 Optionally Display the Map Layout</td>
<td>..............................................</td>
</tr>
<tr>
<td>Step 5 Recompile the Map</td>
<td>.............................................................</td>
</tr>
<tr>
<td>Updating Modified Maps In Dialogs Using ADSC</td>
<td>........................................</td>
</tr>
<tr>
<td>Step 1 Retrieve Dialog XXXDADD</td>
<td>.....................................................</td>
</tr>
<tr>
<td>Step 2 Recompile Dialog XXXDADD</td>
<td>..................................................</td>
</tr>
<tr>
<td>Step 3 Retrieve and Recompile Dialog XXXDUPD</td>
<td>......................................</td>
</tr>
<tr>
<td>Executing the Application</td>
<td>..................................................................</td>
</tr>
<tr>
<td>Optionally Loading the Modified Map</td>
<td>..................................................</td>
</tr>
<tr>
<td>Invoking and Executing the Application</td>
<td>..................................................</td>
</tr>
<tr>
<td>Summary 4</td>
<td>...............................................................................</td>
</tr>
<tr>
<td>Adding Process Logic to a Dialog</td>
<td>..................................................................</td>
</tr>
<tr>
<td>Defining Process Modules Using IDD</td>
<td>....................................................</td>
</tr>
<tr>
<td>Step 1 Invoke the IDD Menu Facility</td>
<td>..................................................</td>
</tr>
<tr>
<td>Step 2 Define Process Module XXXDADD-PREMAP</td>
<td>........................................</td>
</tr>
<tr>
<td>Step 3 Define Process Module XXXDADD-RESPONSE</td>
<td>....................................</td>
</tr>
<tr>
<td>Step 4 Exit from IDD</td>
<td>..........................................................</td>
</tr>
<tr>
<td>Adding Process Modules to Dialogs Using ADSC</td>
<td>........................................</td>
</tr>
<tr>
<td>Step 1 Retrieve Dialog XXXDADD</td>
<td>..................................................</td>
</tr>
<tr>
<td>Step 2 Specify Dialog Options</td>
<td>........................................................</td>
</tr>
<tr>
<td>Step 3 Add a Subschema</td>
<td>..........................................................</td>
</tr>
<tr>
<td>Step 4 Add Process Modules</td>
<td>........................................................</td>
</tr>
<tr>
<td>Step 5 Recompile the Dialog</td>
<td>........................................................</td>
</tr>
<tr>
<td>Correct Errors in Process Modules</td>
<td>..................................................</td>
</tr>
<tr>
<td>Display structural messages</td>
<td>..........................................................</td>
</tr>
<tr>
<td>Display diagnostic messages</td>
<td>..........................................................</td>
</tr>
</tbody>
</table>
Correct structural errors .......................................................... 315
Correct syntax errors ....................................................................... 316
Update dialogs that use the process module ............................................. 317

Test Process Logic ......................................................................... 317
Invoke the Department Application ......................................................... 318
Display Function ADDDEP .................................................................. 318
Test the XXXDADD Dialog ................................................................. 320

Summary 5 ....................................................................................... 322

Modifying Process Logic in a Dialog .................................................... 323
Modifying Process Modules Using IDD ................................................. 324
Step 1 Retrieve the Process Module Definition ............................................. 324
Step 2 Modify Source Statements .......................................................... 325

Updating Modified Process Modules In Dialogs Using ADSC ...................... 328
Step 1 Retrieve and Check Out the Dialog .................................................. 328
Step 2 Recompile the Dialog .................................................................. 330
Execute the Application ...................................................................... 330

Summary 6 ....................................................................................... 331

Defining Work Records Using IDD ....................................................... 332
Instructions 3 .................................................................................... 333
Step 1 Define an Element ....................................................................... 334
Step 2 Define a Work Record ................................................................... 337
Step 3 Specifying Basic Information ......................................................... 337
Adding Elements ................................................................................. 338

Summary 7 ....................................................................................... 340

Completing the Department Application ............................................... 341
Define Process Modules Using IDD ......................................................... 344
Step 1 Define Process Module XXXDUPD-PREMAP .................................. 345
Step 2 Define Process Module XXXDUPD-ENTER ................................... 347
Step 3 Define Process Module XXXDUPD-PA2 ...................................... 353

Completing Dialog XXXDUPD Using ADSC ........................................... 354
Step 1 Retrieve Dialog XXXDUPD .......................................................... 355
Step 2 Add a Subschema ....................................................................... 356
Step 3 Define Dialog Options ............................................................... 357
Step 4 Add a Work Record ................................................................. 358
Step 5 Add Premap and Response Processes ............................................ 359
Step 6 Recompile the Dialog ................................................................. 360

Test the Department Application .......................................................... 361
Invoke the Department Application ......................................................... 361
Display MODDEP Function ................................................................... 362
Verify the Modifications ...................................................................... 363
Display DELDEP Function ................................................................... 366

Summary 8 ....................................................................................... 366
Sample Application Components ................................................................. 368
Development Tools in the CA ADS Environment ........................................ 371
  CA ADS Development Tools ...................................................................... 372
    Invoking Development Tools ................................................................. 373
    Exiting from Development Tools .......................................................... 374
  Using ADSA .............................................................................................. 374
  Using ADSC .............................................................................................. 380
  Using MAPC ............................................................................................. 387
  Using the IDD Menu Facility ...................................................................... 390
Layout Of the DEPARTMENT Record ........................................................... 398

Using ADS Trace .......................................................................................... 399
  Online Trace ............................................................................................. 400
    Building the Trace .................................................................................. 400
    Replaying the Trace ................................................................................ 400
    Environment ........................................................................................... 400
    Flexibility .............................................................................................. 400
      Easy Selection of Options .................................................................... 401
      Trace Options ..................................................................................... 401
      Replay Options ................................................................................... 401
      View Replay Options ......................................................................... 402
    Typical Session ....................................................................................... 402
  CA ADS Trace Access ................................................................................ 403
    Entry Screen ........................................................................................... 404
    Common Fields on Functional Screens .................................................. 404
    Main Menu .............................................................................................. 405
    Conceptual View of CA ADS Trace .......................................................... 405
      Building a Trace .................................................................................. 406
      Recompiling and Executing a Dialog ..................................................... 407
      Replaying the Trace ............................................................................. 407
      Removing the Trace ............................................................................. 407
  Program Function Keys .............................................................................. 407
  Trace Generation ........................................................................................ 409
    Adding EXHIBIT and LITERAL Statements ............................................. 410
    Building Trace Code .............................................................................. 410
    Adding Work Record, Regenerating Dialog, Executing Dialog ................. 410
    Viewing the Replay ............................................................................... 410
    Building Trace Code .............................................................................. 410
    Confirmation ........................................................................................... 411
    Adding Work Record, Recompiling Dialog ............................................. 412
    Executing the Dialog ............................................................................. 412
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Generation Overview</td>
<td>418</td>
</tr>
<tr>
<td>Introduction to Trace Generation</td>
<td>419</td>
</tr>
<tr>
<td>Add Exhibits and Literals</td>
<td>421</td>
</tr>
<tr>
<td>Access CA ADS Trace</td>
<td>421</td>
</tr>
<tr>
<td>Build Trace Code</td>
<td>421</td>
</tr>
<tr>
<td>Add Work Record, Recompile and Execute Dialog</td>
<td>421</td>
</tr>
<tr>
<td>Review Trace Status of Dialogs</td>
<td>421</td>
</tr>
<tr>
<td>Multiple Users</td>
<td>422</td>
</tr>
<tr>
<td>Changing the Trace Option</td>
<td>422</td>
</tr>
<tr>
<td>Specifying Trace Options</td>
<td>422</td>
</tr>
<tr>
<td>Trace Entire Dialog</td>
<td>423</td>
</tr>
<tr>
<td>Trace Processes Within a Dialog</td>
<td>423</td>
</tr>
<tr>
<td>Trace Included Modules</td>
<td>423</td>
</tr>
<tr>
<td>Completing Trace Generation</td>
<td>424</td>
</tr>
<tr>
<td>Change the Trace Option</td>
<td>424</td>
</tr>
<tr>
<td>Trace Selection Tables</td>
<td>424</td>
</tr>
<tr>
<td>Trace On Selections</td>
<td>425</td>
</tr>
<tr>
<td>Trace Off Selections</td>
<td>425</td>
</tr>
<tr>
<td>Trace Generated Only if Command is First Word</td>
<td>425</td>
</tr>
<tr>
<td>DATABASE VERBS ONLY (V)</td>
<td>426</td>
</tr>
<tr>
<td>DEFINE SUBRTN ONLY (D)</td>
<td>426</td>
</tr>
</tbody>
</table>
Using CA ADS

- Using CA ADS Alive (see page 14)
- Using ADS Batch (see page 29)
- Using ADS for IDMS (see page 136)
- Using ADS Trace (see page 399)

Using CA ADS Alive

- Overview (see page 14)
- CA ADS Alive A Powerful Interactive Tool (see page 14)
- Using CA ADS Alive to Debug and Test Dialogs (see page 14)

Overview

This section provides the information needed to run CA ADS Alive. In addition, the many features that CA ADS Alive offers are documented to assist you.

This section includes general information on CA ADS Alive, an online tool that allows CA ADS developers to test dialogs and intercept errors for review and analysis in an online environment. This section provides an overview of the features of CA ADS Alive and provides suggestions for dialog testing.

CA ADS Alive A Powerful Interactive Tool

CA ADS Alive is a source-level testing and debugging tool that provides the CA ADS developer with complete control over the execution of the CA ADS environment. Problems in coding and design can be quickly identified and corrected, and the less experienced developer no longer needs to understand internal data representations or structure of CA ADS runtime control blocks.

In addition, database analysts can use CA ADS Alive to monitor database errors. All CA IDMS status errors are fully interpreted and the information is stored in the CA ADS Alive queue. Database administration personnel can review the information in the queue periodically and use it to analyze database problems.

Using CA ADS Alive to Debug and Test Dialogs

CA ADS Alive is an excellent tool for debugging and testing dialogs. Programmers can use this tool to test dialogs not known to have errors, as well as for debugging dialogs that are not terminating normally.
The extensive capabilities of CA ADS Alive provide the CA ADS programmer with many possibilities for testing. With CA ADS Alive you can intercept and analyze errors, review error interpretation online, and interactively test dialogs. You can then use CA IDMS DME to change CA ADS source online.

Using the features of CA ADS Alive, you can avoid the extra compiles, test data cases, and the need to understand the internal data representations or structure of CA ADS runtime control blocks. This saves both time and resources.

**Dialog Animation**

**Contents**

- Dialog Animation Capabilities (see page 15)
- Two Animation Modes (see page 15)
- Specifying and Altering Stop Locations (see page 16)
- Post-Abort Browse Facility (see page 16)
- Displaying & Modifying Record/Element Contents (see page 16)

The CA ADS Alive online process by which you can test and debug your dialogs is called dialog "animation." Animation is an online view of CA ADS source execution. A typical animation session is described in detail in Section2, CA ADS Alive Session (see page 17).

**Dialog Animation Capabilities**

The CA ADS Alive dialog animation process has several important animation capabilities:

- Two Animation Modes
- Specifying & Altering Animation Stop Locations
- Post Abort Browse Facility
- Displaying & Modifying Record/Element Contents

**Two Animation Modes**

CA ADS Alive provides two animation modes:

- **Non-Interruptible Animation Mode** - Choose this mode if you do not want to specify animation stop (interrupt) points. CA ADS Alive steps through the animation one line of code at a time, pausing for a specified length of time.

  **Note:** The Non-Interruptible Animation Mode causes all CA ADS Alive Animation Runtime Session commands to be inoperative.
Interruptible Animation Mode - Choose this mode if you want to set animation stop (interrupt) points.

Specifying and Altering Stop Locations

Using an EDITOR interface and a combination of primary commands and line specifications, you can specify animation stop locations -- or locations in your source code where the animation process will stop, allowing you to take further action before continuing the animation.

This gives you the ability to control the animation process and quickly and easily pinpoint errors.

Post-Abort Browse Facility

In the event of a dialog execution abort, CA ADS Alive displays a Post-Abort Browse Session of the process containing the error. The line of source that caused the abort is preceded by the associated error message. From the Post-Abort Browse Facility screen, you can expand CA IDMS and LRF messages, display all of the records in the dialog, and transfer control to CA IDMS DME.

You can also save all Post-Abort Browse Facility diagnostic screens to a queue.

The Post-Abort Browse Facility is described in detail in Section 2, CA ADS Alive Session (see page 17). The queue review procedure is described in detail in Section 4, Operations (see page 25).

Displaying & Modifying Record/Element Contents

During the animation session you can also display the contents of a record and its elements.

Suggestions for Testing

There are a number of ways programmers can use CA ADS Alive to test a dialog. Here are some possibilities:

- Test a dialog with an animation stop point at the beginning of each possible dialog path. Execution of the dialog will temporarily halt when the stop point is reached, and you can quickly determine the logic through which the dialog is passing.

- Test a dialog with stop points set at logical points to verify data fields. When data values such as status codes, control fields, or intermediate work fields change during a single execution, you can set stop points to trap and review the changing values.

- Test multiple conditions during one session, avoiding recompilation or regeneration. When data values control the dialog flow, you can set stop points at decision points and alter data values as needed using the RECORD command. In this way, you can exercise alternate dialog paths without creating new test data cases.
- Scroll through a record from beginning to end when the dialog has stopped. You can look for questionable values and change invalid ones before they affect further processing.

- Use the STOP command to set animation stop points to stop only when a field reaches a specific value. This will aid in determining invalid information stored in records or finding error codes when they occur.

- Run a test session in the Non-Interruptible Mode (without animation stop points). In the event of an abort, CA ADS Alive takes you to a browse session from which you can expand error messages, move to an edit session, or go back to an Animation Setup Session (the Animation Setup Session is described in detail in Section 2, CA ADS Alive Session).

---

**CA ADS Alive Session**

This section provides information on using CA ADS Alive.

- **Creating a CA ADS Alive Animation Session** (see page 17)
- Typical CA ADS Alive Screen (see page 20)
- Post-abort Browse (see page 22)
- Record/Element Display & Modification (see page 23)

---

**Creating a CA ADS Alive Animation Session**

The procedure for creating a CA ADS Alive animation session involves selecting a dialog for animation, specifying animation stop locations, and specifying animation session values.

After you complete these tasks you animate the dialog to correct bugs or errors as they occur.

- **Create An Animation Session** (see page 17)
- **Animate the Dialog** (see page 19)

**Create An Animation Session**

To create an animation session, follow these steps:

1. Invoke CA ADS Alive from the CA IDMS/DC system.
   - Type your site-specific task code at the CA IDMS/DC system prompt. By default, the task code is set to ADSALIVE.
   - Press Enter.
   - CA ADS Alive responds by displaying the following Session Specification screen.

   ```
   V81 ENTER NEXT TASK CODE: ADSALIVE
   ```

2. Begin a CA ADS Alive session, and select a dialog from one of the following screens.
   - Session Specification screen:
   - Dialog List Screen
Go to Step 3 or Step 4 for instructions on selecting a dialog from the screen you choose.

3. Select a dialog from the Session Specification screen. The following screen shows an example of wildcard use. This option is specified when CA ADS Alive is installed. Using this method, you may use wildcards (*) in the dialog field. See your DBA if you have any questions about the use of wildcards at your site.
   a. For OPTION, type 1 (Animate).
   b. Type the dictionary, node, dialog, and dialog version number.
   c. Press Enter.

4. Select a dialog from the Dialog List Screen.
   a. From the Session Specification screen, for OPTION type 1 (Animate).
   b. Leave the Dialog and Dialog Version fields blank.
   c. Press Enter to display the Dialog List screen.
   d. Type "S" to specify a Dialog.

5. Specify the animation mode for the interrupt points—the duration that the dialog pauses on a line of code before moving to the next line. CA ADS Alive provides two modes: Interruptible (default) and Non-Interruptible. Choose Non-Interruptible if you do not want to use the default interrupt points.
   To change the mode to Non-Interruptible follow these steps:
   a. From the Specify Animation for DIALOG screen, type N for Interrupt?
   b. For Delay Interval, type the duration in seconds.
   c. Press Enter. A message displays indicating that animation is enabled.
   d. In the COMMAND field, type =3 and press Enter, to proceed.

6. Optional. You can select specific processes for Interruptible Animation.
   a. From the Process List screen, enter "S," "D," or "X" in the column to the left of PROCESS NAME.
      "S" selects every process line.
      "D" deletes the process from animation.
      "X" selects specific process lines. The Animation Setup Edit screen displays for each process selected.
   b. For each process you selected "X," specify the animation mode.

7. Specify stop locations for animation. You set the stop locations using a combination of primary commands and line specifications. For example, the STOP command allows you to stop the animation B-(Before) or A-(At) lines and values after n iterations of a loop, at a numerically repeating interval of an iteration, or combinations.
After you set the stop locations, press Enter.

In the COMMAND filed, type=3 and press Enter, to proceed.

Specify animation session values, from the Animation Session Control screen. The options are:

- **SPECIFY ENVIRONMENT OPTIONS**: Allows you to change the dbname, dbnodc, and dialog version number of the dialog to be animated.

- **SPECIFY TASK STREAM TO INVOKE UPON EXIT FROM SESSION**: Specifies the task code to invoke when you finish the Animation Setup Session. For example, type `ADSdialog-name`. If the dialog is a mainline dialog, CA ADS Alive takes you to the Animation Runtime Session of the specified dialog, as soon as you exit the Animation Setup Session. The dialog does not, however, have to be defined as mainline to use CA ADS Alive. This can be the task code of the application of which your dialog is a part. Leave this field blank to return to the CA IDMS/DC prompt, ENTER NEXT TASK, when you exit the Animation Setup Session.

- **SPECIFY TASK STREAM TO INVOKE UPON EXIT FROM ANIMATION**: Specifies the task code to invoke when you exit from CA ADS Alive during an Animation Runtime Session. For example, type `ADS` to move to the CA ADS main menu as soon as you exit from the Animation Runtime Session. You could also exit to CA IDMS DME by entering DME or the appropriate task code.

### Animate the Dialog

Animate the dialog to correct bugs or errors as they occur.

**To animate the dialog choose one of the following options:**

- Enter =X (Exit) in the COMMAND field of the Animation Session Control screen. If you specified `ADS dialog-name` in the SPECIFY TASK STREAM TO INVOKE UPON EXIT FROM SESSION field of the Animation Session Control screen, you will automatically move to the Animation Runtime Edit screen. Or,

- Type `ADS dialog-name` after the CA IDMS/DC prompt to move to the Animation Runtime Edit screen.

```plaintext
EDIT ---DIALOG AA1 IS RUNNING...          COLUMNS 001 072
COMMAND ===>                      SCROLL ===> PAGE
****** *** TOP OF DATA ************ CA IDMS/ADSAlive ***
==MSG> DICTIONARY:               NODE:
==MSG>   DIALOG: AA1
==MSG>   VERSION: 0001
==MSG>   PROCESS: AA-PREM-P-1
==MSG>   VERSION: 0001
==MSG> ......................NEXT LINE TO EXECUTE FOLLOWS..............
=STOP> MOVE '1' TO AA-ELEMENT-1.
000008 MOVE 99 TO AA-ELEMENT-2.
=STOP> MOVE 'PREMAP COMPLETED' TO AA-ELEMENT-3.
000010 IF AA-ELEMENT-2 = 100
000011 DO.
000012   MOVE '2' TO AA-ELEMENT-1.
000013   MOVE 101 TO AA-ELEMENT-2.
000014 END.
000015 ELSE
000016 DO.
000017   MOVE '3' TO AA-ELEMENT-1.
000018   MOVE 104 TO AA-ELEMENT-2.
```
CA IDMS - 19.0

000019   END.
=STOP> DISPLAY.
****** *** BOTTOM OF DATA ****************************** CA IDMS/ADSAlive ***

Active Commands

BOTTOM
CAPS
CURSOR
DOWN
EDITOR-ID
ENTER
FIND
FIRST
LAST
LEFT
LOCATE
MEMORY
PROFILE
RESET
RESHOW
RFIND
RIGHT
TIME
TOP
UP
ELEMENT
GO
INFO
NOANIMAT
NOPROCSS
NOSTOP
PROCESS
RECORD
REMOVE
REMOVEGO
SHOW
SKIP
STEP
STOP

Non-Interruptible Animation

If you selected non-interruptible animation, you can watch the animation process stop at each line of the dialog. The Non-Interruptible Animation Mode causes all CA ADS Alive Animation Runtime Session commands to be inoperative.

Interruptible Animation

If you selected interruptible animation, animation proceeds to the locations that you set at the Animation Setup Edit screen. The message "NEXT LINE TO EXECUTE FOLLOWS" precedes each animation stop location.

For more information, see Commands (https://docops.ca.com/display/IDMSCU/ADS+Alive+Commands) for detailed descriptions of Animation Runtime Session commands.

Typical CA ADS Alive Screen

The following display shows a typical CA ADS Alive screen. The typical screen includes a COMMAND or OPTION field followed by a message area.
Online documentation is provided for every screen. Enter the HELP command in the COMMAND or OPTION field for screen information.

In this section, the commands that are active are listed after the screen. The commands are described in detail in Common Commands (https://docops.ca.com/display/IDMSCU/Common+Commands).

**Active Commands**

- = DOWN
- END
- EXIT
- HELP
- KEYS
- QUIT
- UP

**Transfer Facility**

An equal sign (=) followed by an alphanumeric string (=string) gives you the ability to transfer to the system-level function represented by string. Below is a list of values for the various system levels.

- Session Specification Screen = 0
- Specify Dialog for Animation Screen = 1
- Session Control Screen = 3
- Exit - Leave Animation Session Intact = X
- Quit - Terminate Animation Session = Q
Post-abort Browse

In the event of an animation/execution abort, CA ADS Alive displays the Post-Abort Browse Session screen showing the process containing the error. The line of source code that caused the abort is preceded by the associated error messages.

This screen is presented if DIAGNOSTIC SCREEN IS YES is specified as a CA ADS statement in your sysgen. See your database administrator if you have any questions.

If ACTIVITY LOG IS YES is specified as a CA ADS statement in your sysgen, the Post-Abort Browse Facility diagnostic screens are saved to a queue. See Section 4, Operations (see page 25) for information on reviewing the CA ADS Alive diagnostic queue.

```
BROWSE -DIALOG AA1 HAS ABENDED.. COMMAND ====> SCROLL ====> PAGE
*** TOP OF DATA ****************************************** CA IDMS/ADSAlive ***
DICTIONARY: NODE:
  DIALOG: AA1
  VERSION: 0001
  PROCESS: AA1-RESPONSE-1
  VERSION: 0001
!THIS RESPONSE Exercises THE NEW ABORT INTERFACE FROM ADS0DEBUG
MOVE 9555 TO SKILL-ID-0455.
MOVE 'VERBALIZATION' TO SKILL-NAME-0455.
MOVE 'SHOOTING THE BREEZE' TO SKILL-DESCRIPTION-0455.
!I FORGOT TO EVER OBTAIN THE SKILL RECORD.
DC173008 APPLICATION ABORTED. BAD IDMS STATUS RETURNED;
STATUS=0809
MODIFY SKILL.
DISPLAY MESSAGE TEXT 'SKILL MODIFIED'.
*** BOTTOM OF DATA ****************************************** CA IDMS/ADSAlive ***
```

**Active Commands**

```
BOTTOM
CAPS
CURSOR
DOWN
EDITOR-ID
ENTER
FIND
FIRST
LAST
LEFT
LOCATE
MEMORY
PROFILE
RESET
RESHOW
RFIND
RIGHT
TIME
TOP
UP
DME
ELEMENT
INFO
RECORD
SETUP
```

From the Post-Abort Browse Session screen, you can use the:
Record/Element Display & Modification

Contents

- Displaying a Record using the RECORD Command (see page 23)
- Displaying a Record using the ELEMENT Command (see page 24)

You can use the RECORD command during an Animation Runtime Edit Session or a Post-Abort Browse Session to display the content of a record and its elements.

You can use the ELEMENT command during an Animation Setup Edit Session to display a list of all records owned by the dialog in which the specified element-name appears.

Displaying a Record using the RECORD Command

To display a record using the RECORD command, enter the RECORD command during an Animation Runtime Session or Post-Abort Browse Session.

If you do not specify a record-name, CA ADS Alive responds by displaying the List of Records screen (shown below). You can then select from all the records owned by the dialog you are animating.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>VERS</th>
<th>DATE CREATED</th>
<th>UPDATED</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA-RECORD-1</td>
<td>0001</td>
<td>mm/dd/yy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AA-RECORD-2</td>
<td>0001</td>
<td>mm/dd/yy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADSO-STAT-DEF-REC</td>
<td>0001</td>
<td>mm/dd/yy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKILL</td>
<td>0100</td>
<td>mm/dd/yy</td>
<td>mm/dd/yy</td>
<td>PUBLIC</td>
</tr>
<tr>
<td><strong>END</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Active Commands

=       DOWN
EXIT    HELP
KEYS    QUIT
UP

For more information, see Section 3, Commands (https://docops.ca.com/display/IDMSCU/ADS+Alive+Commands) for detailed descriptions of the Post-Abort Browse Facility commands.
If you specify a record-name and optionally version, CA ADS Alive responds by displaying the Record/Element Review screen (shown below). See Section 3, Commands (https://docops.ca.com/display/IDMSCU/ADS+Alive+Commands) for detailed information on record/element display and modification commands.

GSI Rnn.nn ----- Record/Element Review ----- CA IDMS DC hh:mm mm/dd/yy
COMMAND ===> GSIRECC0

RECORD: AA-RECORD-1 V 1

DICT:

LINE 0001 OF 0003
05 AA-ELEMENT-1.............................A
05 AA-ELEMENT-2.............................F +000000000
05 AA-ELEMENT-3.............................A

Active Commands

= DISPLAY DOWN END INITIALIZE HELP KEYS QUIT SET AUTOHEX SET HEX/NATIVE SET LOWERCASE SETUP UP

Displaying a Record using the ELEMENT Command

Use the ELEMENT element-name command to display a list of all records in which the specified element-name appears. You can use the ELEMENT command during any Animation Setup Session, Animation Runtime Session, or Post-Abort Browse Session.

CA ADS Alive responds by displaying the List of Records Owning Element screen (shown below).

CA IDMS/ADS Alive Rnn.nn -- List of Records -----------------hh:mm mm/dd/yy
COMMAND ===> USGAESH

Element AA-ELEMENT-1

Enter the "END" command to terminate Inquiry List

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>VERS</th>
<th>DATE CREATED</th>
<th>DATE UPDATED</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA-RECORD-1</td>
<td>0001</td>
<td>mm/dd/yy</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>END</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Active Commands

= DOWN END EXIT HELP KEYS QUIT UP
You can then view a list of all the records in which the specified `element-name` appears. If you want to modify a record/element, return to an Animation Runtime Edit Session and use the RECORD command to access the record.

For more information see Section 3, Commands (https://docops.ca.com/display/IDMSCU/ADS+Alive+Commands) for detailed information on record/element display and modification commands.

```
GSI Rnn.nn -------- Record/Element Review -------- CA IDMS DC hh:mm mm/dd/yy
COMMAND ==> GSIREC0
```

```
RECORD: AA-RECORD-1 V 1

05 AA-ELEMENT-1..............................A
05 AA-ELEMENT-2..............................F +0000000000
05 AA-ELEMENT-3..............................A

Active Commands
```

```
= DISPLAY DOWN END EXIT INITIALIZE HELP KEYS QUIT SET AUTOHEX SET HEX/NATIVE SET LOWERCASE SETUP UP
```

CA ADS Alive Operations

This section describes the CA ADS Alive operating environment, online documentation print facility, and customization macro. The Post-Abort Browse Facility diagnostic queue is also discussed.

CA ADS Alive operates in the following environments and operating systems:

- CA IDMS/DC
- CA IDMS UCF
- z/OS
- z/VSE
- z/VM

**Note:** CA ADS Alive should not be installed in production environments, since some of the buffer modification facilities could lead to security breaches unless stringently secured.
Online Documentation Print Utility

The Online Documentation Print Utility provided with CA ADS Alive allows error messages and other product information to be printed upon request. The Target or Distribution source library member GSIPRINT (z/OS), TOOLJCL library member GSIPRINT.S (z/VSE), or the GSIPRINT EXEC (z/VM), downloaded from the CA ADS Alive installation media, contains the JCL to execute the Online Documentation Print Utility. The online documentation modules for CA ADS Alive processing are listed in the following table.

The printed version of the online documentation is presented one screen per page and includes page reference indexes for screen options. Characters highlighted in the online documentation appear bolded in the printed version.

**Note:** The characters "{ }", which are used to denote optional statements in online documentation modules, appear as "& &" when printed with the Online Documentation Print Utility. The character "<|>", used to denote "or" in online documentation modules, appears as "::" when printed with the Online Documentation Print Utility.

- **COMMANDS**
  - General Browse Commands
- **USGAMEN**
  - CA ADS Alive Tutorial Main Menu
- **USGADLS**
  - Dialog List screen
- **USGAESH**
  - List of Records Owning Element screen
- **USGAEKEY**
  - PF Key Values screen
- **USGAPL**
  - Process List screen
- **USGASES**
  - Animation Session Control screen
- **USGASET**
  - Specify Animation for DIALOG screen
- **USGBRWS**
  - Post Abort Browse Facility commands
- **USGGNRC**
  - Common commands
CA ADS Alive Customization Macros

Two customization macros are provided with CA ADS Alive which allow you to change various CA ADS Alive runtime options:

The CA ADS Alive customization macro gives the system administrator the ability to:

- Alter the task code used to invoke CA ADS Alive.
- Specify the dictionary, node, and version number of online documentation modules.
- Specify whether dialog wildcards are allowed when selecting a dialog at the Session Specification screen.
- Specify whether Non-Interruptible Animation Mode is allowed.
- Specify the number of days that CA ADS Alive queue records should be retained.
- Indicate whether or not the Post Abort Browse screen should be displayed when the Activity Log setting is Yes.

The CA IDMS/DC Sort subroutine customization macro gives the system administrator the ability to:

- Specify the amount of main storage and auxiliary storage to be made available to the CA IDMS/DC Sort subroutine used by CA ADS Alive.
- Indicate how space is to be allocated to buffers at runtime. The allocation of buffers also depends on the record length in a particular sort.

These runtime options can be changed at any time after initial product installation. The source statements are in the USGTPARM and TPSPARM members of your CUSTOM.SRCLIB and the link statements are in the USGTPARM and TPSPARM members of your CUSTOM.LNKLIB.

Reviewing the Post-Abort Browse Queue

Contents

- The Queue Review Procedure (see page 28)
The Queue Review Screen (see page 28)

If ACTIVITY LOG IS YES is specified as an ADSO statement in your sysgen, all Post-Abort Browse Facility diagnostic screens are saved to a queue.

The Queue Review Procedure

You access the Queue Review Facility through CA IDMS/DC. Simply enter the task code QREVIEW and CA ADS Alive responds by displaying the Queue Review screen shown below. The Queue Review screen lists each error for which screens are stored. If there are many entries, you may need to view more than one Queue Review screen. To review or delete an entry, move the cursor to the left-most position of the line for the entry. Enter an S to select the entry for review; enter a D to delete the entry.

When you delete an entry, all screens stored for the entry are deleted from the queue permanently.

Note: The autotask USGADEL automatically deletes from the queue all entries that are older than the number of days specified in the runtime customization macro.

The Queue Review Screen

The following is a description of the Queue Review screen:

- **Release Number**
  The number of the CA ADS Alive release being executed appears before the screen name on the first line.

- **DATE**
  The date on which the error occurred appears in the left column below the screen name. The format is yyddd (Julian date), where yy is the year (such as 99 for 1999) and ddd is the sequential day (such as 001 for January 1st or 033 for February 2nd).

- **TIME**
  The time at which the error occurred appears in the second column below the screen name. The format is hh:mm:ss.tt, where hh is the hour (such as 14 for 1400 hours or 2 p.m.) and mm:ss.tt is minutes, seconds, and hundredths of seconds past the hour (such as 21:08.02 for 21 minutes, eight and two-hundredths seconds).

- **PROGRAM**
  The name of the CA ADS dialog in which an error occurred appears in the third column below the screen name.

- **USER**
  The CA IDMS user ID of the user signed on when the error occurred appears in the fourth column below the screen name. This field is blank if no user was signed on.

- **LTERM**
  The logical terminal ID of the terminal being used when the error occurred appears in the fifth column below the screen name.
STATUS
The CA IDMS error status code of the error that occurred (ADSO for CA ADS dialogs) appears in
the right column below the screen name.

Options Available
The options available to you at this time appear on the last line of the screen:

- **S** - Select an entry for review.
- **D** - Delete an entry.
- **PF8/PF20 key** - Scroll forward one screen.
- **PF7/PF19 key** - Scroll backward one screen.
- **PF3/PF15 key** - Exit the Queue Review Facility.

The PF8/PF20 key can be used to scroll forward when there is more than one Queue Review screen of
entries.

<table>
<thead>
<tr>
<th>DIAGNOSTIC</th>
<th>QUEUE Rnn.nn</th>
<th>QUEUE REVIEW</th>
<th>PROPERTY OF CA, INC</th>
</tr>
</thead>
<tbody>
<tr>
<td>USO0011I</td>
<td>NO MORE IN QUEUE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATE</th>
<th>TIME</th>
<th>PROGRAM</th>
<th>USER</th>
<th>LTRM</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>99182</td>
<td>14:35:47.53</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO1</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>14:53:04.96</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO1</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>15:35:16.93</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO2</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>15:35:44.70</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO2</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>15:36:20.98</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO2</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>15:43:41.09</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO3</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>17:00:29.19</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO1</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>17:01:17.84</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO1</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>17:02:47.46</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO1</td>
<td>ADSO</td>
</tr>
<tr>
<td>99182</td>
<td>18:47:40.26</td>
<td>AA1</td>
<td>BRASC02</td>
<td>VTAMLTO2</td>
<td>ADSO</td>
</tr>
</tbody>
</table>

ENTER-(S)ELECT OR (D)ELETE DETAIL PF7-PRIOR PAGE PF8-NEXT PAGE PF3-EXIT

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 89).
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.

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
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.
IDMSDB—Input and Output Files and File Maps

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 40) and *Application Structure* (see page 46).

**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 70)). 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 39)
- Log File Prefix (see page 39)
- Archiving (see page 40)

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 70) and Control Statements (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 81).

The following topics are discussed below:
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 ddbname IDMSNAP file associated with the linkname IDMSNAP 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 81).

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 41)
- Batch Control Events (see page 41)
- Batch Response Field Values (see page 42)
- Batch Dialog Execution (see page 42)

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 55) and "Response Process Definition Screen" in Section 5.

Testing Batch Control Events

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

```plaintext
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.
IDMSDB--Batch Dialog Execution

**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.
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.

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 46)
- Application Flow of Control (see page 47)
- Accessing Input Files with Multiple Record Layouts (see page 48)

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.

For more information on flow of control, see Runtime Flow of Control (https://docops.ca.com/display/IDMS19/Runtime+Flow+of+Control).

**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

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.
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>Global Records</th>
<th>Page 1 of 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application name: TESTAPP1 Version: 1</td>
<td></td>
</tr>
<tr>
<td>Record name</td>
<td>Version</td>
</tr>
<tr>
<td>1. ADSO-APPLICATION-GLOBAL-RECORD</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
</tr>
<tr>
<td>8.</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

<table>
<thead>
<tr>
<th>Task Codes</th>
<th>Page 1 of 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application name: TEST1 Version: 1</td>
<td></td>
</tr>
<tr>
<td>Task Code</td>
<td>Function</td>
</tr>
<tr>
<td>1. DOIT</td>
<td>F1</td>
</tr>
<tr>
<td>2. DOIT2</td>
<td>FUNC3</td>
</tr>
<tr>
<td>3. UNKOLQ</td>
<td>IUADOLQF</td>
</tr>
</tbody>
</table>
### Response/Function List Screen

#### Sample Screen

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

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

Enter F1=Help  F3=Exit  F4=Prev  F5=Next  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: For more information on the Task Codes screen, see the CA ADS Reference Section.
**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     Drop response (/) _
Response name:      QUIT
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
                           9. Invoke nosave      10. Link nosave
```
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 33).

- **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
Description . . . UNDEFINED
Drop function (/) _
Associated dialog . . . . D2
User exit dialog . . . . . .

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
```

⚠️ **Note:** For more information on the Function Definition screen, see the *CA ADS Reference Section*. 
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 33).

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
Dialog Compiler in Online Mode

Contents

- Main Menu Screen (see page 58)
- Map Specifications Screen (see page 59)
- Database Specifications (see page 60)
- Options and Directives Screen (see page 60)
- Records and Tables Screen (see page 61)
- Process Modules Screen (see page 62)

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

```
Add  Modify  Compile  Delete  Display  Switch

 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

Copyright (C) yyyy CA

Command ===>
Enter F1=Help  F3=Exit  F10=Action
```
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 JPKTD10 Version 1

Map name . . . . __________ Input map . . . . __________
Version . . . . ______ Version . . . . ______
Label . . . . . ______
Paging options _ 1. Wait
2. No Wait
3. Return
Output map . . . __________
Version . . . . ______
Label . . . . . ______
Paging mode . . _ Update
_ Backpage
_ Auto display
Suspense file label ______

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.
CA IDMS - 19.0

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 . . . . . . . . . .

NAME1

SQL Compliance

_ 1. ANSI-standard SQL
2. FIPS

. . . . . . . . .

Date Default Format . . . . . . .
Time Default Format . . . . . . .

Enter

F1=Help

F3=Exit

F4=Prev

_ 1. ISO
_ 1. ISO

2. USA
2. USA

3. EUR
3. EUR

4. JIS
4. JIS

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
Message prefix

JPKTD10

Version

1

. . . . . . . . . . . DC

Autostatus record . . . . . . . . . . ADSO-STAT-DEF-REC

13-Jan-2018

60/442


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**

```
Process Modules

Dialog NAME1 Version 1

Name ____________________________ Type
Version __________ Execute on errors
Key _____ Value ____________________ Drop

Name ____________________________ Type
Version __________ Execute on errors
Key _____ Value ____________________ Drop

Name ____________________________ Type
Version __________ Execute on errors
Key _____ Value ____________________ Drop

Name ____________________________ Type
Version __________ Execute on errors
Key _____ Value ____________________ Drop

* 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.

⚠️ **Note:** For more information on the Process Modules screen, see the *CA ADS Reference Section*. 
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, FIELD A 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 66)
- Extended Map Definition Screen (see page 66)
- File Field Selection Screen (see page 67)
- File Field Edit Screen (see page 68)

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: MAP DEFINITION FIELD SELECTION ADDITIONAL RECORDS
_ EXTENDED MAP DEFINITION _ FILE 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.

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

FOR EXTERNAL FILE ACCESS
RECORD NAME: JMAOUT VER: 1
FILE NAME: JMAOUT VER: 1

MSG PREFIX: DC DECIMAL POINT IS COMMA (Y/N): N
WRITE CONTROL CHARACTER: UNLOCK KEYBOARD.....: Y RESET MDT: Y
**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.

```
CA IDMS/DC FILE MAPPING REL nn.n  *** FILE FIELD EDIT ***

MAP: JMAOUT  VER: 1

EXTERNAL FIELD
FIELD : LAST-NAME  OCCURS:
OF REC : JMAOUT  VER: 1
FILE-PIC: X(15)

INTERNAL FIELD
DFLD : LAST-NAME  OCCURS:
OF REC : JMAOUT  VER: 1

EDIT TABLE:  VER:  (LINK Y/N)  (VALID Y/N/D-DICT)
CODE TABLE:  VER:  (LINK Y/N)  (EDIT Y/N)
FOR INPUT Y  (DATA Y/N)
<EDIT MODULE:  (WITH EDIT B/A/N)>
FOR OUTPUT Y  (DATA Y/N/E)
```
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](#).

  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](#).

  - **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 (}
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.
    If 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 38). 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.
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 an 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:

```
// DLBL SYSTCL, 'idms.sysctl', 0, SD
// DLBL ciln, 'idms.cilib'
// EXTENT , nnnnnn
// ASSGN CL, SEARCH=(ciln), TEMP
// DLBL ADSLOGA, 'log.file'
// EXTENT sysnnn, VOLSER, ,start-track, end-track
// ASSGN sysnnn, DISK, VOL=VOLSER, SHR
// DLBL PAYROLL, 'payroll.file'
// EXTENT sysnnn, VOLSER, ,start-track, end-track
// ASSGN sysnnn, DISK, VOL=VOLSER, SHR
// DLBL REPFILE, 'report.file'
// EXTENT sysnnn, VOLSER, ,start-track, end-track
// ASSGN sysnnn, DISK, VOL=VOLSER, SHR
```

```
CA IDMS - 19.0

// EXEC IDMSFILE,SIZE=(ADSBATCH,128K)
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 ADSBATCH 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, ADSBATCH)
   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
   ADSBATCH 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 81).
   
   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:

   // DBLB ARCFILE,'log.file'
   // EXTENT sysnnn,VOLSER,,,start-track,end-track
   // ASSGN sysnnn,DISK,VOL=VOLSER,SHR
   // DBLB SUSFILE,'payroll.file'
   // EXTENT sysnnn,VOLSER,,,start-track,end-track
   // ASSGN sysnnn,DISK,VOL=VOLSER,SHR
   // DBLB REPFILE,'report.file'
   // EXTENT sysnnn,VOLSER,,,start-track,end-track
   // ASSGN sysnnn,DISK,VOL=VOLSER,SHR
   @SYSFILES
   @SYSCTL
   // DLB SYSIDMS,'#SYSIPT'
   // EXEC IDMSFILE,SIZE=(AUTO,128K)
   @SYSPARMS
   /*
   DEFINE CHARACTERISTICS FOR FILE ADSLOGA
   LRECL=133 RECFM=FIXUNB
   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 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:

```
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 ARCFILE.
DIALOG ARCD5 OUT FIL REPFILE .
ENTRY DIALOG ARCDM.
/*
0028
0029
0458
0459
0450
/*
```
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 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 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 FIXUNB/FIXBLK/FIXED/VARUNB/VARBLK/VARIABLE/UNDEFINED**
  Specifies the record format of the file, as follows:
  - **FIXUNB** specifies that the record format is fixed unblocked.
  - **FIXBLK** specifies that the record format is fixed blocked.
  - **FIXED LENGTH AND BLOCKED/UNBLOCKED** specifies that the record format is fixed blocked or unblocked. The default is BLOCKED.
  - **VARUNB** specifies that the record format is variable unblocked.
  - **VARBLK** specifies that the record format is variable blocked.
  - **VARIABLE LENGTH AND BLOCKED/UNBLOCKED** specifies that the record format is variable blocked or unblocked. The default is BLOCKED.
  - **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 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:

```
ACSept
   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
   .
```
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 a 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 81)
- DSECT for the Log File Prefix (see page 82)
- ADS Batch Control Statements (see page 83)
- z/OS JCL for ADSOBPLG (see page 85)

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

```plaintext
***********************************************************************
***                     BLP: BATCH LOG PREFIX DSECT                    ***
***                     DATE/TIME (KEY)                            ***
***********************************************************************
BLP DSECT 10:41:34 mm/dd/yy
BLPDTK DS CL8 DATE/TIME (KEY)
BLPDATE DS CL4 DATE (00YYDDDC)
BLPTIME DS CL4 TIME (1/10000 SEC)
BLPPGMAM DS CL8 PROGRAM/DIALOG NAME
BLPTYPE DS X RECORD TYPE
SPACE
* THE FOLLOWING ARE VALID VALUES FOR BLPTYPE
  * 0001 DEBUG
  * 0002 USER MESSAGE
  * 0003 EDIT ERROR RECORD
  * 0004 WRITE TO OPERATOR MESSAGE
  * 0005 ABEND DUMP TEXT
  * 0006 STATISTICS
  * 0007 HIGHEST VALID TYPE
  * 0008 IF ON, LOG RECORD IS COMPRESSED
  * 0009 LEN OF LOG REC THAT FOLLOWS PREFIX
  * 0010 LENGTH OF USER ID
  * 0011 RESERVED
  * 0012 LENGTH OF FIXED PORTION OF PREFIX
  * 0013 START OF USER ID
EJECT
```

**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:

```
ADSOBPLG (z/OS)

//ADSOBPLG EXEC PGM=ADSOBPLG,REGION=1024K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.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 SYSOUT=A 

<table>
<thead>
<tr>
<th><strong>idms.dba.loadlib</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:

```
//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)**

```
// DLBL SYSCTL,'idms.sysctl',1999/365,SD  
// DLBL SYSIDMS,'#SYSIPT'  
// DLBL ciln,'idms.cilib'  
// EXTENT ,nnnnn  
// LIBDEF CL,SEARCH=(ciln),TEMP  
```
CA IDMS - 19.0

// 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 75).

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=dictdb,F-NAME=idms.appldict.dictdb,SHARED-UPD=*YES
    /ADD-FILE-LINK L-NAME=dloddb,F-NAME=idms.appldict.dloddb,SHARED-UPD=*YES
    /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></td>
<td>ARCD Writes an employee record to the archive file, then returns control to ARCD1</td>
</tr>
<tr>
<td></td>
<td>ARCD Writes the coverage records associated with an archived employee record to the archive file, then returns control to ARCD1</td>
</tr>
<tr>
<td></td>
<td>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 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>
<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>Record</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REPORT-HDR1</td>
<td>Describe the heading and detail lines for the report</td>
</tr>
<tr>
<td>REPORT-HDR2</td>
<td></td>
</tr>
<tr>
<td>REPORT-DTL</td>
<td></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**

```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**

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

Include within file IDD-INFILE1.

**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**

```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**

```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.
```
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

External field Internal field

INPUT-ID.........................INPUT-ID

**MAP2**

Internal records: DEPARTMENT

OFFICE

External file: ARCFILE

External record: ARCHIVE-RECORD-EMP

External field Internal field

ARCHIVE-TYPE.....................$RESPONSE
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.

<table>
<thead>
<tr>
<th>Dialog</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCD</td>
<td>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</td>
</tr>
<tr>
<td>ARCD</td>
<td>Writes the specified employee record to the archive file, then returns control to ARCD1</td>
</tr>
</tbody>
</table>
Dialog Description

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.                                                    *
!**********************************************************************
```

**ARCD1-PM**

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

**ARCD1-RESPONSE**  
(default RP)

**Work records**

<table>
<thead>
<tr>
<th>WORK-RECORD1</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORT-HDR1</td>
</tr>
<tr>
<td>REPORT-DTL</td>
</tr>
</tbody>
</table>

**Subschema**

<table>
<thead>
<tr>
<th>EMPSS01</th>
</tr>
</thead>
</table>
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 *
!*-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'.
  IF DB-EOF
    WRITE LOG TEXT '***EOF ON INPUT***'.
    LEAVE APPLICATION.
  END.
!
!*************************************************
!*-PASS CONTROL TO DLG ARCD2, WHICH ARCHIVES *
!* THE EMPLOYEE RECORD.                      *
!*-PASS CONTROL TO DLG ARCD3, WHICH ARCHIVES *
!* THE ASSOCIATED COVERAGE RECORDS.           *
!*************************************************
LINK NOSAVE 'ARCD2'.
LINE NOSAVE 'ARCD3'.
%****************************************************%
!*-SET UP FOR TRANSACTION REPORT, THEN PASS       *
!* CONTROL TO DLG ARCD4, WHICH, ALONG WITH       *
!* ARCD5, WRITE A REPORT LINE.                   *
%****************************************************%
MOVE 'RECORD ARCHIVED' TO REPORT-STATUS.
MOVE EMP-ID-0415 TO REPORT-ID.
MOVE EMP-LAST-NAME-0415 TO REPORT-LNAME.
MOVE EMP-FIRST-NAME-0415 TO REPORT-FNAME.
LINK NOSAVE 'ARCD4'.

%****************************************************%
!*-DELETE THE EMPLOYEE RECORD FROM THE           *
!* DATABASE ALONG WITH ALL COVERAGE RECORDS.      *
!*-ON ANY DB ERROR, TERMINATE THE APPL.          *
%****************************************************%
OBTAIN CALC EMPLOYEE.
ERASE EMPLOYEE ALL MEMBERS.
IF DB-ANY-ERROR
   ABORT TEXT 'DB ERROR ON EMPLOYEE ERASE'.

%****************************************************%
!*-READ THE NEXT INPUT RECORD.                   *
%****************************************************%
READ TRANSACTION.

%****************************************************%
!*SUBROUTINE ERRRTN                              *
!*-CALLED WHEN THE REQUESTED EMP REC IS NOT      *
!* FOUND IN THE DATABASE                         *
!*-SET UP FOR TRANSACTION REPORT, THEN          *
!* PASS CONTROL TO 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 'RECORD NOT FOUND' TO REPORT-STATUS.
MOVE EMP-ID-0415 TO REPORT-ID.
MOVE SPACES TO REPORT-FNAME.
MODIFY MAP TEMP FOR (INPUT-ID) EDIT ERROR.
WRITE TRANSACTION.

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.
!*-RETRIEVE THE EMPLOYEE'S OFFICE AND
```
Dialog ARCD2: Mapout Operation

Internal field | External field
$RESPONSE .................>ARCHIVE-TYPE
DEPT-ID-0410 ...........>ARCHIVE-DEPT-ID
OFFICE-CODE-0450 .......>ARCHIVE-OFFICE-CODE
ARCHIVE-EMPLOYEE-RECORD ......>ARCHIVE-EMPLOYEE-RECORD

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**

```
!**********************************************************************
!*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
!*----------------------------------------------------------------------
!
!*************************************************
!*-MOVE RECORD TYPE TO $RESPONSE.          *
!*************************************************
MOVE 'C' TO $RESPONSE.
!
!*************************************************
!*-RETRIEVE A COVERAGE RECORD.          *
!*-ON END-OF-SET, RETURN TO ARCD1.       *
!*-IF ANY OTHER DB ERROR, ABORT APPL.    *
```

**Map name: MAP3**

Ddname: ARCFILE

**Work records**

None

**Subschema**

EMPSS01
IF FIRST-TIME
   OBTAIN FIRST COVERAGE WITH EMP-COVERAGE.
ELSE
   OBTAIN NEXT COVERAGE WITHIN EMP-COVERAGE.
IF DB-END-OF-SET
   RETURN.
IF DB-ANY-ERROR
   ABORT TEXT 'DB ERROR ON COVERAGE OBTAIN'.
!
!*************************************************
!*-MOVE COVERAGE RECORD TO ARCHIVE RECORD        *
!*************************************************
MOVE COVERAGE TO ARCHIVE-COVERAGE-RECORD.
!
!*************************************************
!*-WRITE THE RECORD, THEN REEXECUTE THE         *
!* PROCESS TO ARCHIVE THE NEXT COVERAGE REC.    *
!*************************************************
WRITE TRANSACTION CONTINUE.

Dialog ARCD3: Mapout Operation

Internal field        External field
$RESPONSE......................>ARCHIVE-TYPE
ARCHIVE-COVERAGE-RECORD........>ARCHIVE-COVERAGE-RECORD

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:

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.
ENDIF

!**************************************************************
!*-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:

```
IDMSDB--Dialog ARCD5
```

The premap process and mapout operation are shown below.

**Dialog ARCD5: Premap Process**

```plaintext
!*PREMAP
!*ARCD5-PM
!*  -EXECUTED AT THE BEGINNING OF DLG ARCD5.
!*  -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>ARCFILE</td>
<td>ARCFILE</td>
<td>VB</td>
<td>128</td>
<td>3000</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, ARCFILE, 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.

INFILE1

3000
4000
5001

REPFILE1
### 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.

#### Table: Employee Records

<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, RECORD ARCHIVED</td>
</tr>
<tr>
<td>4000</td>
<td></td>
<td>RECORD NOT FOUND</td>
</tr>
<tr>
<td>5001</td>
<td>WATSON</td>
<td>BRIAN, RECORD ARCHIVED</td>
</tr>
</tbody>
</table>

#### SUSFILE1

4000

#### ARCFFILE

E4000012300030SEPH STERMS 100 HANGER DRIVE BOSTON ...
C800101830102M232
C830103506061M232
E40000125001BRIAN WATSON 300 ST CATHERINE W. MONTREAL ...
C850101 D123

#### ADSLOGA

IDMS DC506801 V1 SUSPENSE FILE SUSFILE1 RECORD# 1 IMAGE IS '4000'
***EOF ON INPUT***
IDMS DC507001 V1 BASE FILE INFILE1 CLOSED.
IDMS DC507003 V1 3 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 ISUSFILE1 CLOSED.
IDMS DC507004 V1 1 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 ARCFFILE CLOSED.T
IDMS DC507004 V1 2 LOGICAL RECORDS WRITTEN.
IDMS DC507001 V1 BASE FILE ARCFFILE CLOSED.E
IDMS DC507004 V1 3 LOGICAL RECORDS WRITTEN.
IDMS DC507002 V1 BASE DATASET MEN.C0600.MPK.ARCFFILE CLOSED.
IDMS DC507004 V1 1 PHYSICAL RECORDS WRITTEN.
IDMS DC507001 V1 BASE FILE REPFILE1 CLOSED.
IDMS DC507004 V1 5 LOGICAL RECORDS WRITTEN.
IDMS DC507002 V1 BASE DATASET MEN.C0600.MPK.REPFILE1 CLOSED.
IDMS DC507004 V1 1 PHYSICAL RECORDS WRITTEN.
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

<table>
<thead>
<tr>
<th>Component</th>
<th>Name</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responses</td>
<td>R2</td>
<td></td>
</tr>
</tbody>
</table>
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.

Note: The process modules shown in Step 3 have embedded comment lines, indicated by an exclamation point (!) in column 1. You do not have to key in these lines.
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 1</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 2</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 3</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 4</td>
<td>Write transaction report lines to an output file.</td>
</tr>
<tr>
<td>ARCD 5</td>
<td></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**
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

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

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 DIALOOGS 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 SENSPE
  !- 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
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**

```
RESD2-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.
*RESD2-RESPONSE ISSUES A CONTINUE COMMAND AFTER AN

Map name: MAP2
Ddname: ARCFILE

RESD2-RESPONSE
(default RP)

Work records
WORK-RECORD1
INPUT-RECORD1

Subschema
EMPSS01
```
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 WORK-STATUS = 'RESTORED'
RETURN.

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

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

READ TRANSACTION.

SUBROUTINE RESTRTN
-CALLED WHEN A MATCH IS FOUND.
-SET CURRENCY ON EMPLOYEE'S OFFICE AND
DEPARTMENT, WHOSE IDS WERE ARCHIVED ALONG
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.
******************************************************************************

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:

---

**Mapin**

Map name: MAP3
Ddname: ARCFILE

**Response process**

RESD3-RESPONSE (default RP)

**Work records**

WORK-RECORD1

**Subschema**

EMPSS01

---

IDMSDB--Dialog RESD3

The mapin operation and response process are shown below.

**Dialog RESD3: Mapin Operation**

External field | Internal field
---|---
ARCHIVE-TYPE | $RESPONSE
ARCHIVE-COVERAGE-RECORD | ARCHIVE-COVERAGE-RECORD

**Dialog RESD3: Response Process**

!**********************************************************************
!*RESD3-RESPONSE*
!* EXECUTED AFTER A COVERAGE RECORD HAS BEEN READ FROM THE ARCHIVE*
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.
**Employee-Record Report Application**

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

**Sample Output**

<table>
<thead>
<tr>
<th>FC10001</th>
<th>FICTIONAL COMPANY, INC.</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE</td>
<td>EMPLOYEE LISTING BY DEPARTMENT</td>
<td>11/05</td>
</tr>
<tr>
<td>EMP ID</td>
<td>NAME--</td>
<td>STATUS</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>0479</td>
<td>CLOTH</td>
<td>TERRY</td>
</tr>
<tr>
<td>0329</td>
<td>FINN</td>
<td>PHINEAS</td>
</tr>
<tr>
<td>0469</td>
<td>KASPAR</td>
<td>JOE</td>
</tr>
<tr>
<td>4014</td>
<td>SHEPARD</td>
<td>LISA</td>
</tr>
<tr>
<td>0355</td>
<td>TIME</td>
<td>MARK</td>
</tr>
<tr>
<td>0439</td>
<td>WILCO</td>
<td>ROGER</td>
</tr>
<tr>
<td>0479</td>
<td>CLOTH</td>
<td>TERRY</td>
</tr>
<tr>
<td>0329</td>
<td>FINN</td>
<td>PHINEAS</td>
</tr>
<tr>
<td>0469</td>
<td>KASPAR</td>
<td>JOE</td>
</tr>
<tr>
<td>4014</td>
<td>SHEPARD</td>
<td>LISA</td>
</tr>
<tr>
<td>0355</td>
<td>TIME</td>
<td>MARK</td>
</tr>
<tr>
<td>0439</td>
<td>WILCO</td>
<td>ROGER</td>
</tr>
</tbody>
</table>

DEPARTMENT 5200 TOTAL EMPLOYEES: 61

DEPARTMENT 3100 TOTAL EMPLOYEES: 10

TOTAL EMPLOYEES ALL DEPARTMENTS: 61

Employee-record Flow Example

The diagram below shows the file access, components, and flow of control for the application.
IDMSDB--Employee-Record Report 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPT</td>
<td>Acts as the mainline routine for the application, reading database records and passing control to other dialogs as required; writes department and report total lines, as shown below:</td>
</tr>
<tr>
<td></td>
<td>DEPARTMENT 3100 TOTAL EMPLOYEES: 10</td>
</tr>
<tr>
<td></td>
<td>TOTAL EMPLOYEES ALL DEPARTMENTS: 61</td>
</tr>
<tr>
<td>EDIT</td>
<td>Writes department header lines, as shown below:</td>
</tr>
<tr>
<td></td>
<td>DEPARTMENT ID: 5200 ...</td>
</tr>
<tr>
<td>DEPT</td>
<td>Writes detail column header lines, as shown below:</td>
</tr>
<tr>
<td></td>
<td>EMP ID ──NAME─...</td>
</tr>
<tr>
<td>HDR</td>
<td>EMP ID ──NAME─...</td>
</tr>
<tr>
<td>COL</td>
<td>EMP ID ──NAME─...</td>
</tr>
<tr>
<td>MHD</td>
<td>EMP ID ──NAME─...</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-REPFILE1, 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</td>
</tr>
<tr>
<td></td>
<td>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).
03 WK-DATE-X       PICTURE IS X(6)  REDEFINES WK-DATE-HOLD.
03 WK-DATE-WORK.
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.

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

**HDRMAP**

<table>
<thead>
<tr>
<th>Internal records:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>External file:</td>
<td>IDD-REPFILE1</td>
</tr>
<tr>
<td>External record:</td>
<td>REPT-HEADER</td>
</tr>
</tbody>
</table>

<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**

<table>
<thead>
<tr>
<th>Internal records:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>External file:</td>
<td>IDD-REPFILE1</td>
</tr>
<tr>
<td>External record:</td>
<td>EMPL-DETAIL</td>
</tr>
</tbody>
</table>

<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 DDDDL 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>Writes department header lines to the report file</td>
</tr>
<tr>
<td>DEPT</td>
<td>Writes department header lines to the report file</td>
</tr>
<tr>
<td>HDR</td>
<td>Writes detail column header lines to the report file</td>
</tr>
<tr>
<td>COL</td>
<td>Writes detail column header lines to the report file</td>
</tr>
<tr>
<td>MHD</td>
<td>Writes detail lines to the report file</td>
</tr>
<tr>
<td>R</td>
<td>Writes detail lines to the report file</td>
</tr>
<tr>
<td>DLINE</td>
<td>Writes page header lines to the report file</td>
</tr>
<tr>
<td>NE</td>
<td>Writes page header lines to the report file</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**

```plaintext
*---------------------------------------------------------------------
!*EMPL-REPT-EDIT
!*---------------------------------------------------------------------
IF FIRST-TIME
   DO.
      MOVE DATE TO WK-DATE-HOLD.      !*SET UP CURRENT DATE
      CALL CVRTDATE.
      MOVE WK-DATE-WORK 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 'LTD' 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.
       MOVE 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.

***-SET UP DEPARTMENT TOTAL EMPLOYEE COUNT LINE TO*!
"DEPARTMENT XXXX TOTAL EMPLOYEES: ZZZ9"
***-LOOK AS FOLLOWS:
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

13-Jan-2018 128/442
!*SUBROUTINE TO EDIT AN EMPLOYEE COUNT TOTAL LINE*
!*SUBROUTINE TO EDIT AN EMPLOYEE COUNT TOTAL LINE*
!*SUBROUTINE TO EDIT AN EMPLOYEE COUNT TOTAL LINE*
DEFINE TOTEDIT.

MOVE 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        *
!*SUBROUTINE TO PUT OUT PAGE HEADER LINES        *
!*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      *
!*SUBROUTINE TO CENTER JUSTIFY HEADER TITLES      *
!*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 YYMMD TO  *
!*SUBROUTINE TO CONVERT A DATE FROM YYMMD TO  *
!*SUBROUTINE TO CONVERT A DATE FROM YYMMD TO  *
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

<table>
<thead>
<tr>
<th>Internal field</th>
<th>External field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL-CHAR......</td>
<td>&gt;CTRL-CHAR</td>
</tr>
<tr>
<td>LEFT............</td>
<td>&gt;LEFT</td>
</tr>
<tr>
<td>CENTER..........</td>
<td>&gt;CENTER</td>
</tr>
<tr>
<td>RIGHT...........</td>
<td>&gt;RIGHT</td>
</tr>
</tbody>
</table>
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 dialog definition]

**Premap process**

DEPT-HDR-EDIT

**Mapout**

Map name: HDRMAP
Ddname: EMPLRPT

**Work records**

WORK-PRINT-EDIT
DEPARTMENT

**Subschema**

EMPSS01

Dialog DEPTHDR: Premap Process

```
/*DEPT-HDR-EDIT
DEPT-HDR-EDIT*

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

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**

```plaintext
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:
IDMSDB--Dialog DLINEOUT

The premap process and mapout operation are shown below.

**Dialog DLINEOUT: Premap Process**

```
!*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:

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**

```
Map name: DTLMAP
Ddname: EMPLRPT
```

```
WORK-PRINT-EDIT
```

```
EMPSS01
```
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:

DEPARTMENT ID: 5200 THERMOREGULATION

<table>
<thead>
<tr>
<th>EMP ID</th>
<th>NAME</th>
<th>STATUS</th>
<th>SOC SEC</th>
<th>START DATE</th>
<th>TERM DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0479</td>
<td>CLOTH</td>
<td>ACT</td>
<td>028701666</td>
<td>11/02/79</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0329</td>
<td>FINN</td>
<td>ACT</td>
<td>011787878</td>
<td>06/16/79</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0469</td>
<td>KASPAR</td>
<td>LOA</td>
<td>036768888</td>
<td>06/05/82</td>
<td>00/00/00</td>
</tr>
<tr>
<td>4014</td>
<td>SHEPARD</td>
<td>UNK</td>
<td>000000000</td>
<td>00/00/00</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0355</td>
<td>TIME</td>
<td>ACT</td>
<td>034560128</td>
<td>05/05/84</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0439</td>
<td>WILCO</td>
<td>ACT</td>
<td>111000023</td>
<td>11/11/79</td>
<td>00/00/00</td>
</tr>
</tbody>
</table>

DEPARTMENT 5200 TOTAL EMPLOYEES: 6

DEPARTMENT ID: 3100 INTERNAL SOFTWARE

<table>
<thead>
<tr>
<th>EMP ID</th>
<th>NAME</th>
<th>STATUS</th>
<th>SOC SEC</th>
<th>START DATE</th>
<th>TERM DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0024</td>
<td>DOUGH</td>
<td>ACT</td>
<td>022337878</td>
<td>08/08/76</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0029</td>
<td>GALLWAY</td>
<td>ACT</td>
<td>014567777</td>
<td>10/10/81</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0003</td>
<td>GARFIELD</td>
<td>JENNIFER</td>
<td>ACT</td>
<td>021994516</td>
<td>01/21/77</td>
</tr>
<tr>
<td>0028</td>
<td>GRANGER</td>
<td>PERCY</td>
<td>011234545</td>
<td>05/27/83</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0027</td>
<td>HEAROWITZ</td>
<td>VLADIMIR</td>
<td>ACT</td>
<td>031896154</td>
<td>09/09/89</td>
</tr>
<tr>
<td>0020</td>
<td>JACOBI</td>
<td>JAMES</td>
<td>018813465</td>
<td>11/11/89</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0019</td>
<td>JENSEN</td>
<td>JULIE</td>
<td>033456718</td>
<td>09/29/82</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0035</td>
<td>LITERATA</td>
<td>LARRY</td>
<td>023567831</td>
<td>09/09/80</td>
<td>00/00/00</td>
</tr>
<tr>
<td>0023</td>
<td>O'HEARN</td>
<td>KATHERINE</td>
<td>ACT</td>
<td>019556712</td>
<td>05/04/78</td>
</tr>
</tbody>
</table>
Using ADS for IDMS

This section is intended for the following users:

- New application developers who need to learn how to use CA ADS
- Experienced application developers who are unfamiliar with CA ADS
- Any application developers who want to look up basic steps for defining application components

For more information, see the following topics:

- Building a Prototype (see page 136)
- Designing Maps (see page 142)
- Designing Dialog (see page 147)
- ADS Naming Conventions (see page 162)
- Performance Considerations 1 (see page 166)
- Overview of CA ADS Application Development (see page 177)
- Defining an Application Structure Using ADSA (see page 182)
- Defining a Screen Display Using MAPC (see page 213)
- Defining Dialogs Using ADSC (see page 239)
- Modifying the Application Structure Using ADSA (see page 257)
- Modifying a Map Using MAPC (see page 265)
- Adding Process Logic to a Dialog (see page 285)
- Modifying Process Logic in a Dialog (see page 323)
- Defining Work Records Using IDD (see page 332)
- Completing the Department Application (see page 341)
- Sample Application Components (see page 368)
- Development Tools in the CA ADS Environment (see page 371)
- Layout Of the DEPARTMENT Record (see page 398)

Building a Prototype

The development of a prototype can be approached in a variety of ways, depending upon the needs of the design team. The procedures suggested in this section are based on a three-stage approach:

1. The initial stage performs rudimentary navigation of the application
2. The second stage begins to perform data retrieval and update

3. The final stage incorporates refinements that reflect the more complex requirements of an application running in a production environment

Each stage of the prototype is discussed below.
- Stage 1 Building the Basic Prototype (see page 137)
- Stage 2 Adding Process Logic and Data Retrieval (see page 140)
- Stage 3 Refining the Maps and Processes (see page 142)

Stage 1 Building the Basic Prototype

Contents
- Compiling the Application (ADSA) (see page 138)
- Compiling the Maps (MAPC) (see page 138)
- Compiling the Dialogs (ADSC) (see page 139)
- User Review (see page 140)

First Stage

You can develop the first stage of the prototype quickly and easily because only skeletal maps and dialogs are needed for execution by the CA ADS runtime system. Typically, you compile maps with just enough information to identify their use in the application process, and one dialog is compiled for each map. The dialogs do not need a premap process or a response process. With a minimum of time and effort, you can see how the application is going to work even before data processing takes place.

Load Modules Needed

To build an executable prototype, you need to provide load modules for the runtime system by:

- Compiling the application -- The application and its components (the functions and responses) are defined and compiled with ADSA.

- Compiling the maps -- Each map is formatted, defined, and compiled with MAPC.

- Compiling the dialogs -- Each dialog is identified, associated with the appropriate map, and compiled with ADSC.

The prototype can be executed when the application, map, and dialog load modules are available for use by the CA ADS runtime system. At this point, you have a meaningful version of the prototype that can be presented for user review and modification.

Each of the activities for building the basic prototype is discussed separately below, followed by user review considerations.
Compiling the Application (ADSA)

The amount of detail you provide for a prototype can be as extensive as you like, but the basic prototype does not have to be elaborate.

Steps in Compiling an Application

After signing on to ADSA, you can compile an application as follows:

1. Identify the application -- The name of the application and related information are supplied on the Main Menu

2. Name the task code -- The task code that designates an entry point into the application Task Codes screen. If there are multiple entry points, each task code must be defined individually.

3. Define the responses -- The responses that initiate the functions of an application are defined on the Response/Function List and the Response Definition screens. Each response is defined on a separate screen.

4. Define the functions -- The functions that are initiated by the responses are defined on the Response/Function List and the Function Definition screens. Each function is defined on a separate screen.

5. Compile the application -- The application is compiled by selecting the compile option from the Main Menu.

Application Definition Block and Task Activity Table

When the above-named activities are completed successfully, ADSA defines an Application Definition Block (ADB) for the application and updates the Task Activity Table (TAT). Both the ADB and the TAT are stored as load modules in the dictionary and are used by the CA ADS runtime system when the application is executed.

Compiling the Maps (MAPC)

Steps in Compiling a Map

Maps that are compiled for the first stage of the prototype usually contain all literal fields. You sign on to MAPC and take the following steps to produce the prototype screens:

1. Identify the map -- The map name and related information are supplied on the Main Menu.

2. Format the screen -- The map design can be painted automatically using the autopaint facility of MAPC, or can be explicitly laid out using the Layout screen. The extent of the map design is left up to you. Some developers indicate the purpose of the screen with a one-line caption.
2. Other developers prefer to format a screen that more closely resembles the final application version, but with literal values (such as hyphens or underscores) assigned to the variable data fields.

3. Edit the map fields -- Each field on the map can be edited using the Field Definition screen. If all fields have been defined as literals, you can accept the default attributes by pressing [Enter] and proceed to the next editing screen; MAPC requires editing for each field on the map.

4. You can take the defaults in many areas, but you must specify element names for fields. Whenever you request COMPILE, if adequate information is available, defaults are taken and the map is compiled.

5. Compile the map -- You compile the map by selecting the compile option on the Main Menu.

A map load module is stored in the DDLDCLOD area of the data dictionary when the map has been compiled successfully.

Compiling the Dialogs (ADSC)

Using ADSC

One dialog needs to be compiled for each map used by the prototype. To compile a prototype dialog, sign on to ADSC and specify the dialog name on the Main Menu.

In addition, specify the subschema (if there is one) and the map associated with the dialog on the Database Specifications and Map Specifications screens.

Because these are map-only dialogs, there is no need to use any other ADSC screens for the first stage of the prototype. Compile the dialog by selecting the compile function from the Main Menu. Each dialog is defined on a separate screen.

Considerations

The following considerations should be noted when compiling a dialog:

- If a dialog is defined as a function on the Response/Function List screen in ADSA, it must be defined on the Main Menu screen in ADSC (using the same dialog name).

- If a dialog is associated with a task code, it must be defined as a mainline dialog.

- The associated map must be compiled before the dialog can be compiled.

Fixed Dialog Block

ADSC defines a Fixed Dialog Block (FDB) for every dialog that is compiled successfully. The FDB is stored as a load module in the dictionary and is used by the CA ADS runtime system when the application is executed.
User Review

With the creation of the dialog load module, the basic prototype is ready to be presented to the user for online review. Modifications should be made to the existing prototype, the necessary load modules recompiled, and the prototype resubmitted for review until the users are satisfied.

Stage 2 Adding Process Logic and Data Retrieval

Contents

- ADSA Enhancements (see page 140)
- Populating the Data Dictionary (IDD) (see page 141)
- MAPC Enhancements (see page 141)
- ADSC Enhancements (see page 141)

Second Stage

The prototype becomes more functional in the second stage. You can add activities such as the following to the prototype:

- Global records (ADSA)
- Security restrictions such as signon menus (ADSA)
- Display capabilities (MAPC and IDD)
- Premap and response process logic (ADSC and IDD)

The ADSA, MAPC, ADSC, and IDD activities used for these enhancements are described separately below.

ADSA Enhancements

You can add the following ADSA features to the prototype at this point:

- Global records (that is, records that are available for use by all dialogs in the application) can be defined on the Global Records screen.

  Note that the ADSO-APPLICATION-GLOBAL-RECORD appears on this screen and is always included automatically as a global record.

- User-program records (that is, records that are to be passed to a user program) can be defined on the Function Definition (Program) screen, if they are needed.

- Valid responses listed for a function can be resequenced or their display can be suppressed on the second screen of the Function Definition (Menu) screen.
Signon menu functions can be specified on the second page of the General Options screen. When security is designated as REQUIRED or OPTIONAL on this screen, the following steps should be taken:

- Name a function as the application’s signon function, using the second page of the General Options screen.
- Specify the above-named function as a menu function, using the Function Definition screen.
- Specify that the menu is a signon menu, using the Menu Specification screen.
- Specify the SIGNON system function as the function initiated by a response, using the Response/Function List screen.
- Specify the response that initiates the SIGNON system function as a valid response for the named menu function, using the Function Definition (Menu) screen.

When these changes have been made, recompile the application.

Populating the Data Dictionary (IDD)

The data dictionary must contain the following components if they are to be used by the application:

- Dialog premap and response processes -- Premap and response processes must be stored as process modules in the data dictionary. If premap or response processes are associated with a dialog, process modules must be defined in the data dictionary before the dialog can be compiled. Modules are added to the dictionary with the MODULE statement specifying LANGUAGE IS PROCESS.

- Map records and dialog work records -- All work records used by a dialog and all records associated with maps must be defined in the data dictionary before the dialogs and maps can be compiled. Similarly, an application cannot be compiled unless all global records associated with the application are defined in the data dictionary. Records are added with the RECORD statement.

- Edit and code tables -- All stand-alone edit and code tables associated with map records must be defined in the data dictionary before the map is compiled. Edit and code tables are added with the TABLE statement.

For complete details on adding process modules, records, and tables to the data dictionary, refer to the CA IDMS IDD Quick Reference Section.

MAPC Enhancements

Variable map fields that were specified as literals for the first stage of the prototype should be redefined as data fields and edited accordingly.

When the appropriate enhancements have been made, the map should be recompiled.

ADSC Enhancements

You now use ADSC, recompiling the dialog to include the premap and response processes, as well as the changes made to the map associated with this dialog. After signing on to ADSC and naming the appropriate dialog on the Main Menu screen, you can add the following features:
Stage 3 Refining the Maps and Processes

Third Stage

The final stage of development can focus on refinement of the map design and the map field attributes. Some of the following additions can be made:

- Incorporate additional fields in the maps
- Add or change map field attributes
- Specify automatic editing on selected map fields
- Provide informational messages
- Add error messages

Designing Maps

Maps displayed during the execution of the application interface directly with the terminal operator and, therefore, can influence the success of an application. Consequently, you must consider the appearance of the menu screens and the layout of the dialog maps.

Successful Map Design

A successful map design should exhibit the following attributes:

- Consistency -- Entities (for example, fields, headings, labels, responses, messages, and control keys) should have the same meaning or effect throughout the application. The meaning or effect need not be identical for every map, but should be consistent within the broader confines of the system. In general, there are two special fields on any screen: a message field and a response code field. These areas should appear in a constant location on the screen throughout any application; for maximum effectiveness, they should remain standard for all applications at a site.
Convenience -- Features of the system should be designed to associate related entities by using similar constructs, positioning, and responses to produce similar reactions from the system. For example, assign one particular control key to initiate the update function in all the dialogs of a given application.

Supportiveness -- The reactions of the system should enable the user to handle normal contingencies conveniently. Tutorial aids should be available when needed. Displayed informational and/or error messages should be meaningful.

The remainder of this section discusses the following aspects of map design:

- Standards to consider when designing maps
- Mapping procedures that can be adopted by an installation
- Choices available in the design of menu maps
- Suggestions for designing dialog maps

Design Standards for a Dialog Map

You need to consider the following standards when designing dialog maps:

- Design the map with the terminal operator in mind. For example, a very dense screen is tiring and difficult to use. In general, the screen most pleasing to the eye is about 40% full.

- The placement of fields on the screen, the use of high intensity, and the neatness of the format have a great deal of impact on the effectiveness of the system.

- When the screen is sent to the terminal, the cursor should be in the position most likely to be used for data entry. Other frequently used fields should be easily accessible with the tab and return keys.

- The sequence of fields, when tabbed, should match the most common pattern used for data entry.

- Fields requiring special attention should be highlighted and clearly visible.

- The screens should be as uncluttered as possible. The common error of using one screen format for excessive and/or dissimilar functions tends to produce cluttered or busy screens; separate screens with some common fields are more usable.

- Terminal users should be able to initiate processing by typing in the necessary data and pressing a control key. They should not be required to make decisions that could have been incorporated in program logic, nor should they be forced to use control keys or responses needlessly.

Mapping Procedures

The following illustrate the mapping procedures that might be implemented by a specific site:
Have one individual (for example, the data administrator) responsible for creating and modifying all maps.

As much as possible, use the features of MAPC to handle editing, error handling, error messages, and modifying field attributes.

Use a standard map template. Whenever possible, keep data fields in columns and double space rows of data.

Use the BRIGHT attribute to contrast items on the screen that have different uses (for example, highlight required fields). Be consistent in the use of attributes.

Use the cursor in a consistent manner. For example, either place the cursor at the first field to be used for data entry or at the field where the terminal operator is to enter the next function.

Use the BRIGHT attribute for redisplaying data fields that are in error.

Choosing Menu Map

When designing an application, you need to decide if system- or user-defined menu maps are to be used. The system-defined menu provides a standard format for the information provided by the developer during the definition of the functions and responses of the application in an ADSA session. If a format other than the standard format is desired (for example, you want to redefine certain literal fields on the map or want to supply site-specific headers), the user-defined menu map is used. Both types of maps are discussed separately below.

Contents

- System-Defined Menu Maps (see page 144)
- User-Defined Menu Maps (see page 145)
- Reformatting and Recompiling the System-Defined Menu (see page 145)
- Designing a Menu/Dialog (see page 146)

System-Defined Menu Maps

Menu Formats

If the menu map is to be system-defined, you have the option of using one of the following menu formats:

- Short description menu map (ADSOMUR1) -- The menu screen that lists 30 valid menu responses per page; a short (12-byte) textual description is displayed for each response.

- Long description menu map (ADSOMUR2) -- The menu screen that lists 15 valid menu responses per page; a long (28-byte) textual description is displayed for each response name.

- Signon menu map (ADSOMSON) -- The menu screen that requires a CA IDMS/DC or DC/UCF validation of user ID and password before the menu request can be processed. The standard signon menu map can have 12 valid menu response names per page with 28 bytes of descriptive text displayed for each.
User-Defined Menu Maps

When user-specific modifications to the existing system-defined menu maps are necessary, you can change the menu maps by using either of the following techniques:

- Reformatting and recompiling the standard system-defined menu
- Designing a menu/dialog (that is, a menu map that is part of a menu/dialog function)

Each of these methods is discussed below.

Reformatting and Recompiling the System-Defined Menu

You can reformat and recompile the existing system-designed menu map, retaining the same name. This method allows you to use the standard menu function rather than designing and using a menu/dialog function.

Steps to Reformat the System Menu

Take the following steps to reformat the system menu:

1. Obtain the source for the map being used (that is, ADSOMUR1, ADSOMUR2, or ADSOMSON) from the source data sets created when the distribution tape was installed. The maps are stored as members under their own names.

2. Use the batch mapping compiler to store the source in the data dictionary.

3. Use MAPC to modify and recompile the menu map.

Considerations

When recompiling a menu map with MAPC, observe the following rules:

- ADSO-APPLICATION-MENU-RECORD is a required map record. Optionally, the menu can map to additional records, but it must always map to the ADSO-APPLICATION-MENU-RECORD.

- The menu must contain the same number of responses per page as the number of responses for the selected map (that is, 30 for ADSOMUR1, 15 for ADSOMUR2, or 12 for ADSOMSON).

- The AMR-RESPONSE field of the ADSO-APPLICATION-MENU-RECORD record is a required field. The first response name on the map must map to the first occurrence of AMR-RESPONSE. Each subsequent response name must map to the next corresponding occurrence.

- The AMR-USER-ID and AMR-PASSWORD fields of the ADSO-APPLICATION-MENU-RECORD are required on a signon menu map. The user ID data field must map to AMR-USER-ID, and the password data field must map to AMR-PASSWORD.
Designing a Menu/Dialog

You can design and compile an entirely new menu with MAPC. This map must be defined as a menu/dialog function of the application.

Steps for Defining a Menu/Dialog Function

Follow these procedures when defining a menu/dialog function:

1. Design and compile the map using MAPC. Observe the following rules when compiling the map:
   - ADSO-APPLICATION-MENU-RECORD must be one of the records associated with the map.
   - The AMR-RESPONSE field is required for all menus. The number of required occurrences depends on the number of responses per page (to a maximum of 50) specified on the ADSA Menu Specification screen. The first response name on the map must map to the first occurrence of AMR-RESPONSE; each subsequent occurrence must map to the next corresponding occurrence of AMR-RESPONSE.
   - The AMR-USER-ID and AMR-PASSWORD fields are required for signon maps. The user ID data field must map to AMR-USER-ID, and the password data field must map to AMR-PASSWORD.
   - All other fields on the ADSO-APPLICATION-MENU-RECORD are optional. The map data fields used must be associated with the appropriate fields on the record (for example, heading data must map to AMR-HEADING).

2. Add the process source to the data dictionary in an IDD session. (The dialog associated with the menu does not have to include any process code, although the choice of a menu/dialog function suggests that some processing is intended.)

3. Compile the dialog in an ADSC session, associating the map and any processes with the dialog using the ADSC Map Specifications screen. Note that the dialog must be compiled to include the map before the application can be executed at runtime.

4. Define the dialog as a menu/dialog function for the application, using the ADSA Response/Function List screen.
An installation can develop standard map templates and the associated boilerplate code for site-specific menu/dialogs. When a menu is needed, programmers can obtain a copy of the template/boilerplate, fill in the appropriate fields and the edit/code tables needed for those fields, and submit it to the data administrator for approval.

### Designing Dialog Maps

**Design Questions**

Each dialog map is associated with its own dialog and must be designed to reflect the function of the associated dialog. The application specifications developed during the initial design stages can be used to answer design questions such as the following:

- How many of the dialogs specified for this application will require maps?
- What premap and response processes are required for each map?
- What job is performed by each process?
- Will the map be used to pass data between processes and/or between dialogs? What data will be passed?
- What database and mapping work records are associated with the map?
- What editing criteria should apply to the map fields?

**Map Templates**

Just as site-specific standards can be established for menu/dialogs, an installation can use map templates to standardize the formatting of maps associated with dialog functions. Programmers can obtain a copy of the template; fill in the appropriate fields, indicating the corresponding map record fields; and submit this information to the data administrator. The data administrator can then add the necessary map design, map records, and edit/code tables (if any) to the data dictionary.

### Designing Dialog

A dialog is a unit of work within a CA ADS application that enables interaction with the terminal operator. Because dialogs are the basic building blocks of a CA ADS application, it is important that they be well-designed. This section discusses characteristics and design features of dialogs that merit the attention of application developers.

### Dialog Characteristics

The characteristics of a dialog determine its role within the application. Each dialog has an implicit level and status, and can pass and receive control of the processing. The significance of the dialog level and status and the manner in which control is passed are discussed below.
Dialog Level

The level of a dialog refers to its position within the application structure. You can pass processing control to a dialog at the next lower level, the same level, the next higher level, or the top level of the application structure.

⚠️ **Note:** The meaning of TOP changes whenever a LINK command is executed. The dialog issuing LINK becomes the current TOP.

Impact of the Dialog Level

At runtime, the dialog level affects the following aspects of an application:

- **Availability of data** -- When combined with the manner in which processing control is received, the level of a dialog governs the data passed in the record buffer blocks and the currencies that are established, saved, stored, or released.

- **Use of system resources** -- The runtime system maintains record buffer blocks, database currency blocks, and variable dialog blocks for dialogs at each level. There is a direct correlation between the number of dialog levels in an application and the size of the storage pool that is needed.

- **Performance** -- The number of dialog levels can affect the performance of an application. For example, performance times are affected if a frequently accessed dialog is located three or four levels down in an application structure.

An application can be composed of any number of dialog levels, but the most efficient application uses many levels only when absolutely necessary.

Mainline Dialog

The top-level dialog must be a **mainline** dialog and must be defined as such by the application developer. A mainline dialog is the entry point to the application. An application can have more than one mainline dialog; entry points can also be established at a lower level in the application structure. In addition to defining a task code for the top-level dialog, the developer can identify an alternative entry point by using the Task Codes screen to associate a task code with a lower-level function.

Dialog Status

Operative and Nonoperative Status
A dialog can have an **operative** or a **nonoperative** status within the application thread. A dialog becomes operative when it receives control and begins executing; at a given level, only one dialog can be operative at a time. When control passes to a dialog at another level, the issuing dialog can remain operative or can become nonoperative, depending upon the level of the next dialog. For example, when control is passed with the LINK command, the issuing dialog remains operative; when control is passed with the TRANSFER command, the issuing dialog becomes nonoperative.

As long as a dialog is operative, all data that it has acquired is retained. When a dialog becomes nonoperative, its data is released. See "Database currencies" later in this section for a summary of the way in which a dialog’s status is affected by the successful execution of a control command.

**Application Thread**

Within the application structure, only one dialog executes at a time. The sequence of dialog execution within an application structure is called the **application thread**. The response of the terminal operator determines the dialogs that constitute a given application thread. The following diagram shows an application structure and one application thread.

One dialog can exist in several places within the application structure and be part of the same or different application threads. A dialog can execute more than once within the application thread whether or not it remains operative.

In the diagram below, the shaded boxes represent an application thread that includes dialog A, dialog C, and dialog D.
Dialog Control

A dialog passes control to another dialog based on the execution of a control command and/or the terminal operator’s selection of processing. The dialog that receives control can be a different dialog, a copy of the executing dialog, or all or part of the executing dialog itself.

Operations Performed by Control Commands

You can use specific control commands to perform the following operations:

- Pass processing control from one dialog to another dialog or to a user program
- Display a dialog’s map
- Terminate an existing dialog or application
- Exit CA ADS
- Pass processing control to specified points within a dialog and reinitialize the record buffers associated with a dialog

Most of the control commands used are available to all applications. When designing dialogs that will become part of an ADSA application, you can also use the EXECUTE NEXT FUNCTION command.

ADS - Design Considerations

You need to keep the following CA IDMS/DB, CA IDMS/DC and DC/UCF, and CA ADS features in mind when designing the dialogs:

- Database Currencies (see page 150)
- Database, Work, and Map Records (see page 151)
- Logical Records (see page 154)
- NEW COPY Records (see page 154)
- Working Storage Areas (see page 154)
- Queue Records (see page 155)
- Scratch Records (see page 155)
- Extended Run Units (see page 156)
- Longterm Locks (see page 157)
- Global Records (see page 158)

Each of these issues is presented below.

Database Currencies

In CA ADS, currency is maintained automatically for the user. To facilitate this feature, a currency control block is created that maintains currency information. At runtime, a currency block is created for each dialog in the application structure that performs database requests.
Passing Currencies

Database currencies are passed from one dialog to another dialog at a lower level, enabling dialogs to continue database processing from an established position in the database. Currencies are cumulative. The currencies established by each dialog are passed to lower-level dialogs, which, in turn, establish their own currencies; the cumulative currencies are passed to the next lower-level dialog.

Currencies are established, saved, restored, and released as follows:

- Established -- Currency is established with the dialog’s first functional database call. Established currencies are updated when database commands (for example, FIND, OBTAIN, ERASE) are encountered during the transaction. Currency is nulled when a dialog receives control with a RETURN or TRANSFER command.

- Saved -- When a LINK, DISPLAY, or INVOKE command is issued, the database currencies established with the last database command in the dialog are saved. Saved currencies are available to lower-level dialogs and are restored to the issuing dialog if processing control returns.

- Restored -- Saved currencies are restored when CA ADS opens a transaction in the dialog receiving control (that is, saved currencies are restored just prior to the first database call).

- Released -- When a LEAVE, RETURN, or TRANSFER command is issued, all database currencies at the same and lower levels are released. The dialog receiving control must establish its own currencies or use the currencies passed to it from another higher-level dialog.

Note that currencies, as described in this section, pertain only to DML run units. SQL run units are not managed by the CA ADS runtime system. Please refer to the CA IDMS SQL Programming Section for more information on SQL programming techniques.

The following table shows the ways in which the passing and receiving of control affects the contents of the currency block.

Record buffer management is affected by whether the record used by the dialog is a database, work, or map record; a logical record; or a record that has been assigned the NEW COPY attribute. The manner in which the CA ADS system allocates space for these records in the Record Buffer Block (RBB) is discussed below.

Database, Work, and Map Records

At the beginning of each application thread, the CA ADS runtime system allocates a primary Record Buffer Block (RBB) and initializes a buffer in the RBB for each record associated with the top-level dialog.

Considerations

All lower-level dialogs can access records in any of the existing buffers, unless one of the following conditions is true:

- The dialog that receives control accesses a database record or a work record that has been assigned the NEW COPY attribute during dialog compilation.
The dialog that receives control accesses a database record, work record, or logical record not used by a higher-level dialog.

The dialog that receives control accesses a database record that uses a subschema not used by a higher-level dialog.

If one or more of these conditions exist, CA ADS allocates and initializes an additional buffer for the record.

Additional Buffers

Additional buffers are also allocated and initialized when one of the following situations exists:

- The record is assigned the WORK RECORD attribute during dialog compilation.
- The record is associated with the map used by the dialog.
- The record is named explicitly in a database command.

The following diagram shows the sequence in which CA ADS initializes record buffers as a series of dialogs receives control.

When dialog A begins executing, CA ADS allocates buffers for the EMPLOYEE and SKILL record types. Dialog B uses the previously allocated EMPLOYEE record buffer, but requires a new buffer for the OFFICE record. Dialog C requests and receives a new copy of the EMPLOYEE record buffer, but uses the previously allocated SKILL record buffer. Dialog D requires new buffers for both the DEPARTMENT and JOB records. CA ADS allocates a secondary RBB to accommodate the DEPARTMENT record, but uses the remaining space in the primary RBB for the JOB record.
IDMSDB--Database, Work, and Map Records
Logical Records

Considerations

When an application thread contains dialogs that use a combination of database records and logical records, special considerations apply with respect to record buffer management. For each database record component of a logical record, CA ADS initializes individual, contiguous record buffers. The logical record components are placed in the buffer in the order named in the logical record definition.

For example, consider the EMP-JOB-LR logical record, which consists of four database records: EMPLOYEE, DEPARTMENT, JOB, and OFFICE records. If dialog B accesses EMP-JOB-LR, CA ADS initializes new record buffers for each of the four records listed above (in that order) regardless of whether buffers for one or more of the records were initialized when dialog A, a higher-level dialog, began executing. Therefore, dialog B (and lower-level dialogs accessing the same logical record) does not have access to data established in the record buffer by dialog A. However, dialogs at levels lower than dialog B will use the buffers established by dialog A if those dialogs use the same database records as dialog A.

When using both database records and logical records, the first dialog of the application thread should include an INITIALIZE command for the logical record. This action associates the logical record with the top-level dialog and ensures that the buffer for the entire logical record will be allocated and available to all lower-level dialogs. Lower-level dialogs will use the component record buffers established at the highest level unless the logical record itself is referenced.

NEW COPY Records

Records can be assigned the NEW COPY attribute on the Records and Tables screen during the definition and compilation of a dialog. The NEW COPY designation signifies that the record in question is to receive newly initialized record buffers when the dialog is executed.

The NEW COPY attribute is used when the programmer wants to obtain another occurrence of a record type without overwriting the data that is in the current buffer. To have the use of a second, temporary buffer for the same record type, the programmer links to a lower-level dialog that has specified NEW COPY for that record. An occurrence of the record type is brought into the new buffer and processed as directed. When control returns to the calling dialog, the record buffer at the upper level contains the same data as before; the data in the lower-level record buffer is no longer available.

Dialogs at a level lower than the dialog with a NEW COPY record will not use the NEW COPY buffer, but will use the first buffer allocated for the record.

Working Storage Areas

DC/UCF queue and scratch areas can be used by the CA ADS dialogs as working storage areas. The methods by which dialogs can store and use records in the queue and scratch areas are presented below.
Queue Records

Queue records can be used as work records that are shared by tasks on all DC/UCF terminals. Entries are directed to a queue with database commands embedded in the dialogs or batch programs. Queues can transfer data across the entire DC/UCF system and are maintained across system shutdowns and crashes. Currencies and locks are not passed between tasks.

Characteristics

Queue records have the following characteristics:

- A queue header record is allocated either at system compilation or by an application dialog.
- Queue records participate in a set in the data dictionary; this set is commonly referred to as a queue.
- Queue records are locked by each task; no other task can use them until the locks are released.

Queues created at system compilation with the system QUEUE statement can be accessed by a CA ADS application. Additionally, an application can create its own queues by requesting storage space with a GET QUEUE statement in the dialog process code.

Functions

An application can use queue records to accomplish the following functions:

- Automatically initiate a task -- DC/UCF initiates a task that processes the queue entries when the number of entries in a queue reaches a specified limit or when a specified time interval has passed. For example, an application can write records to a queue and the system will route the records to a printer when the collected records exceed the specified limit.
- Avoid prime time updating -- Records that need to be updated can be collected on a queue; the queue can be accessed by a batch program at a low-use time.
- Prevent run-away tasks -- A maximum limit can be established for the number of entries permitted in a queue. The UPPER LIMIT parameter of the QUEUE statement is especially useful in a test environment to prevent a looping program from filling the scratch/queue area.

For detailed descriptions of the queue management commands, refer to the CA ADS Reference Section.

Scratch Records

Scratch records are shared between tasks and saved across the transactions of a CA ADS application. Used as a temporary storage area, scratch records provide a means of passing data between tasks running on the same terminal; they are not accessible to tasks that execute on other terminals and are not saved across a system shutdown or a system crash.

Characteristics

The following characteristics are associated with scratch records:
Scratch records are stored in the data dictionary.

Multiple scratch areas are allowed for a task and multiple records can be maintained within a scratch area.

Currency is maintained for each area and record, and can be passed between tasks.

The scratch area is allocated dynamically within the storage pool. When all scratch records are deleted, the area will also be deleted.

Functions

Scratch records can be used in the following ways within an application:

- To save input acquired from two or more dialogs over the course of the application.

- To allow multiple occurrences of a record to be mapped out at one time. For example, if the names, addresses, and phone numbers of all department employees need to be mapped onto the same screen in multiples of five, the following steps could be taken:

  1. Walk the set of employee records, moving the required data to a work record that contains multiply-occurring fields.

  2. When the work record contains the data on five employees, move the contents of the work record to the scratch area with a PUT SCRATCH command so that, in effect, a screenful of data on five employees is put on each record in the scratch file.

  3. Walk the set of scratch records when the screens of information are to be displayed.

- To pass the contents of the record buffer when a dialog receives control with a TRANSFER command. Data acquired by the dialog issuing a TRANSFER command is not available to the dialog receiving control. However, the dialog receiving control could access buffer data that had been placed in a scratch record.

Refer to the CA ADS Reference Section for detailed descriptions of the scratch management commands.

Extended Run Units

Typically, a CA ADS transaction begins when the dialog issues a command accessing the database (for example, OBTAIN) and ends when the runtime system encounters the next control command issued by the dialog (that is, LINK, INVOKE, DISPLAY, TRANSFER, LEAVE, or RETURN). An extended transaction is a transaction that is kept open when the runtime system encounters the LINK command under the following circumstances:

- When the LINK is to the premap process of a dialog with no associated subschema

- When the LINK is to the premap process of a dialog with an associated schema and subschema identical to those of the calling dialog

- When the LINK is to a user program

Implications
Implications of the extended transaction are as follows:

- Currencies are passed to the lower-level dialog and are restored upon return to the upper-level dialog.

- Currencies are not passed to user programs; currencies are saved and restored to the upper-level dialog when control is returned.

- The lower-level dialog can perform error checking to decide whether to issue a ROLLBACK command.

- Because a FINISH is not issued, record locks held by the upper-level dialog are not released. A COMMIT can be coded in the upper-level dialog if the developer needs to release locks before linking to the lower-level dialog.

- If a COMMIT is issued prior to the LINK command and an abend occurs in the lower-level dialog, the rollback will be incomplete; the rollback will only go to the COMMIT checkpoint and not to the start of the transaction.

- If a lower-level user program opens its own transaction, a deadlock can occur. The possibility of a deadlock condition can be avoided by taking either of the following actions:
  
  - Issue a COMMIT prior to the LINK.
  
  - Pass the subschema control block to the user program and let the program use the same transaction. Issue no BINDs or FINISHes in the user program.

For more information on the extended transaction, refer to the CA ADS Reference Section.

**Longterm Locks**

The KEEP LONGTERM command sets or releases longterm record locks. Longterm locks are shared or exclusive locks that are maintained across transactions. Once the longterm locks are set, all other transactions are restricted from updating or accessing the named records until the dialog explicitly releases the locks. The following example requests the release of all longterm locks associated with the current task:

```
KEEP LONGTERM ALL RELEASE
```

**Monitoring Database Activity**

The KEEP LONGTERM command can also be used to monitor the database activity associated with a record, set, or area. When a dialog is updating records that could also be updated by another user, the following code can be included in the premap process of the named dialog:

```
KEEP LONGTERM longterm-id NOTIFY CURRENT record-name
```

This command instructs the CA ADS runtime system to monitor the database activity associated with the current occurrence of the named record type.

The following code is included in the response process of the same dialog:
KEEP LONGTERM \longterm-id

TEST RETURN NOTIFICATION INTO return-location-v

This command requests notification of any database activity against records that were specified in the KEEP LONGTERM premap process. If appropriate, the dialog can check the return value placed in the specified work record field.

For more information on the KEEP LONGTERM command, refer to the CA ADS Reference Section.

Global Records

Global records are records that are available to all dialogs, maps, and user programs in an application. Subschema records cannot be defined as global records.

System-Defined Global Record

The ADSO-APPLICATION-GLOBAL-RECORD is the system-defined global record that enables communication between the application and the runtime system. To be accessed by a dialog, the ADSO-APPLICATION-GLOBAL-RECORD must either be specified as a dialog work record or be associated with the dialog's map. This record is initialized when an application is first loaded by the runtime system.

All fields in the ADSO-APPLICATION-GLOBAL-RECORD are addressable by dialogs or user programs. Selected fields from the ADSO-APPLICATION-GLOBAL-RECORD are listed below. For a complete listing of these fields, refer to the CA ADS Reference Section.

AGR-NEXT- FUNCTION

The AGR-NEXT-FUNCTION field contains the name of the next function that is to be executed. When the dialog associated with the current function ends with an EXECUTE NEXT FUNCTION command, the function named in the AGR-NEXT-FUNCTION field is executed by the runtime system. A dialog or user program can query this field to check what the next function will be. Modification of the AGR-NEXT-FUNCTION field, however, does not change the next function to be executed; a change in the next function can only be accomplished by modification of the AGR-CURRENT-RESPONSE field (see below).

AGR-DEFAULT- RESPONSE

The AGR-DEFAULT-RESPONSE field contains the default response value specified on the Function Definition screen when an application is compiled. When a value is specified and the screen includes a data field for a default response, the terminal operator can type in a new value or can space out the value that appears.

AGR-CURRENT- RESPONSE

The AGR-CURRENT-RESPONSE field contains the response specified by the terminal operator. The process code of a dialog or user program can also move values into this field, overwriting the user response. Note that, if AGR-CURRENT-RESPONSE is modified by a dialog, security is not checked for the response moving into the field, even if security is associated with this response.
When EXECUTE NEXT FUNCTION is encountered within process code, the response named in the AGR-CURRENT-RESPONSE field is executed if it is a valid response for the current function. The AGR-CURRENT-RESPONSE field determines the next function in the application thread (that is, it determines the value moved into the AGR-NEXT-FUNCTION field).

**Moving a Value into AGR-CURRENT-RESPONSE**

The following diagram shows the manner in which the runtime system moves a value into the AGR-CURRENT-RESPONSE field. The value in AGR-CURRENT-RESPONSE depends upon whether the AGR-DEFAULT-RESPONSE field contains a value; whether the terminal operator enters a new value in the response field; or whether there is a response value associated with the control key (other than ENTER) pressed by the terminal operator. The runtime system executes the response named in the AGR-CURRENT-RESPONSE field after determining that it is a valid response for the current function.
The AGR-EXIT-DIALOG field initially contains the name of the exit dialog specified on the Application Definition screen. This field can be used to link to a special routine. For example, one department of a company might want the employee name specified as John Doe, while another department wants the name specified as Doe, John. The same dialog could be used for both departments by linking to an exit dialog (that is, LINK TO AGR-EXIT-DIALOG) containing a name routine.

AGR-PRINT-DESTINATION

The AGR-PRINT-DESTINATION field initially contains the default name of the printer for the application as specified on the ADSA Application Definition screen. Dialogs and user programs can use this print destination with the WRITE PRINTER DESTINATION command.

AGR-USER-ID

The AGR-USER-ID field can be queried by dialogs and user programs.

AGR-PRINT-CLASS

The AGR-PRINT-CLASS field initially contains the default printer class for the application as specified on the ADSA Application Definition screen. The dialog can reference this field with the WRITE PRINTER CLASS command.

AGR-SIGNON-SWITCH

The AGR-SIGNON-SWITCH field can be queried to determine if there has been a valid signon.

AGR-SIGNON-REQMTS

The AGR-SIGNON-REQMTS field indicates whether signon is optional, required, or not used for the signon menu, as specified on the Security screen. This field can be referenced for additional security checking.

AGR-MAP-RESPONSE

The AGR-MAP-RESPONSE field can be used as a response field, in place of the $RESPONSE field, in any user-defined nonmenu map. The dialog can initialize this response field before mapout so that the desired default response appears on the map. For input purposes, the AGR-MAP-RESPONSE field works in the same manner as the $RESPONSE field. For information on the $RESPONSE field, refer to the CA IDMS Mapping Facility Section.

AGR-MODE

The AGR-MODE field initially contains the value STEP or FAST as specified on the Application Definition screen. Typically, the design of a dialog map includes a field that displays the value of AGR-MODE. The terminal operator can change this field at any time.

The following examples show how the AGR-MODE field can be used, in conjunction with the EXECUTE NEXT FUNCTION command, to implement a STEP/FAST mode for an ADSA application. The logic in the first example assumes that all data field validation is handled by the automatic editing specifications in the dialog's map. The logic in the second example assumes that additional data validation is required in the response process code. In both cases, any data entered by the terminal operator is always processed. Note that the first pass flag field has no significance in FAST mode.
Example 1

This sample process code illustrates the manner in which a dialog can query the AGR-MODE field of the ADSO-APPLICATION-GLOBAL-RECORD to determine what course to follow. If the dialog is in STEP mode, the dialog redisplay the screen with a confirmation message for the terminal operator; if in FAST mode, control is passed immediately to the next function. The initial value of AGR-MODE is supplied by the runtime system; the user can alter the value of AGR-MODE at any time during application execution.

IF ANY OF (EMPLOYEE-NBR, SKILL-CODE, SKILL-LEVEL) ARE CHANGED DO.
   MOVE 'Y' TO FIRST-PASS-FLAG.
   MOVE EMPLOYEE-NBR TO WK-EMPNBR.
   MOVE SKILL-CODE TO WK-SKLCODE.
   MOVE SKILL-LEVEL TO WK-SKLEVEL.
   LINK TO 'CEMDUEMP'.
END.
IF AGR-STEP-MODE DO.
   IF FIRST-PASS-FLAG='Y'
   DO.
      MOVE 'Y' TO FIRST-PASS-FLAG.
      DISPLAY MSG TEXT IS 'EMPLOYEE UPDATED'.
   END.
   MOVE 'Y' TO FIRST-PASS-FLAG.
   END.
EXECUTE NEXT FUNCTION.

Example 2

The sample code shown in the following example shows the use of the AGR-MODE field when data validation needs to be handled by code in the response process. Note that the EXECUTE NEXT FUNCTION command is never encountered while uncorrected validation errors still exist.

IF ANY OF (EMPLOYEE-NBR, SKILL-CODE, SKILL-LEVEL) ARE CHANGED DO.
   MOVE 'Y' TO FIRST-PASS-FLAG.
   IF EMPLOYEE-NBR GE 2000 AND SKILL-CODE='A'
   DO.
      MOVE 'Y' TO ERROR-FLAG.
      DISPLAY MSG TEXT IS 'EMPLOYEE NUMBER/SKILL CODE MISMATCH'.
   END.
   MOVE 'N' TO ERROR-FLAG.
   MOVE EMPLOYEE-NBR TO WK-EMPNBR.
   MOVE SKILL-CODE TO WK-SKLCODE.
   MOVE SKILL-LEVEL TO WK-SKLEVEL.
   LINK TO 'CEMDUEMP'.
   CALL EMPDTE25.
END.
IF ERROR-FLAG='Y'
   DISPLAY MSG TEXT IS 'EMPLOYEE NUMBER/SKILL CODE MISMATCH'.
   CALL EMPDTE25.
!**************************************************
DEFINE EMPDTE25.
!**************************************************
IF AGR-STEP-MODE DO.
   IF FIRST-PASS-FLAG='Y'
   DO.
      MOVE 'N' TO FIRST-PASS-FLAG.
      DISPLAY MSG TEXT IS 'EMPLOYEE UPDATED'.
   END.
   MOVE 'Y' TO FIRST-PASS-FLAG.
The following fields from the ADSO-APPLICATION-GLOBAL-RECORD are often mapped to screens associated with user-defined nonmenu maps:

- AGR-DIALOG-NAME
- AGR-APPLICATION-NAME
- AGR-CURRENT-FUNCTION
- AGR-FUNCTION-DESCRIPTION
- AGR-DATE
- AGR-USER-ID
- AGR-MODE
- AGR-PASSED-DATA
- AGR-MAP-RESPONSE

For an illustration of how these fields can be used on maps, refer to "Designing maps" in Section 4.

**ADS Naming Conventions**

The establishment of naming conventions reduces the accumulation of redundant data and improves the overall design of an application. Naming convention standards apply to the components of an application as well as to the database entities accessed by the application. Naming conventions for application entities and database information entities are each discussed separately below.

- Naming Application Entities (see page 162)
- Naming Database Information (see page 164)

**Naming Application Entities**

Naming conventions make it easier to keep track of application components as they are created and maintained. While mnemonic names can work well for less complex applications, mnemonics are inadequate when handling the large volume of complex applications that typically exist at most sites. Adhering to a naming convention eases the construction of component names, eases the reconstruction of component names if one is forgotten, and eases the use and maintenance of an application.

**Naming Convention Standards**

The following table lists the naming convention standards used for the sample application in this section.
<table>
<thead>
<tr>
<th>Position</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>CA IDMS product</td>
</tr>
<tr>
<td>2-3</td>
<td>EM</td>
<td>Type of application:</td>
</tr>
<tr>
<td></td>
<td>IS</td>
<td>Employee information</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>Information system</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>Financial system</td>
</tr>
<tr>
<td></td>
<td>SY</td>
<td>Manufacturing system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System activities</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>Component type:</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>Dialog</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>Function</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>Map</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>User-defined program</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>Help module</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>Report</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>Report</td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>Subschema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Menu</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>Component functions:</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Add operation</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Encode/decode (column 4 indicates table)</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Delete operation</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>Edit operation (column 4 indicates table)</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>Inquiry operation</td>
</tr>
<tr>
<td></td>
<td>U</td>
<td>Modify operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update operation</td>
</tr>
<tr>
<td>6-8</td>
<td>xxx</td>
<td>Component designator:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Three characters used as unique identifiers for the component</td>
</tr>
</tbody>
</table>

### Assigning Names

Names in an application can be assigned in the following manner:

- Dialogs, maps, tables, programs, help modules, and reports can use the conventions in preceding table as follows:
  - Dialog: CEMDILIS
  - Map: CEMMILIS
  - Code table: CEMTCLIS
  - Edit table: CEMTELIS
  - Menu: CEMUILIS
  - User program: CEMPILIS
  - Help module: CEMHILIS
  - Report: CEMRILIS
Dialog premap and response process names can be the concatenation of the dialog name and the suffix -PREMAP or -RESPONSE, as in the following examples:

CEMDILIS-PREMAP
CEMDILIS-RESPONSE

If there are multiple response processes, the suffixes can be structured to reflect the function of each response process, as follows:

CEMDILIS-ADDRESP
CEMDILIS-DELRESP

Names for subroutines included in the premap and response processes can be made up of a meaningful name of up to 6 characters with a 2-digit suffix, as follows:

PASSDT05
MESSGE97
DBERR99

The numeric suffixes can be assigned and incremented as the subroutines appear in the dialog. This numbering convention makes it easier to locate a subroutine in the dialog listing. For example, MESSGE97 is located near the end of the listing while PASSDT05 is located near the beginning.

Naming Database Information

Glossary

The creation of a glossary can be an effective means of establishing naming conventions for database information. The glossary can be stored in the dictionary where it is readily available as a reference tool. Tools such as the glossary also aid in the development of consistent site-specific application coding standards.

Example

The following example shows sample entries from one type of glossary. This example shows one way in which a glossary can be defined; each design team must determine the naming conventions that best suit its needs. Note that the word WORD in this example is a user-defined entity defined to the data dictionary, as follows:

ADD CLASS NAME IS WORD
CLASS TYPE IS ENTITY.

ADD WORD ABEND ABBREVIATED NEVER
ADD WORD ABSOLUTE ABBREVIATED NEVER
ADD WORD ACCEPT ABBREVIATED NEVER
ADD WORD ACCOUNT ABBREVIATED SOMETIMES ABBR ACCT
ADD WORD ACCRUAL ABBREVIATED NEVER
ADD WORD ACCUMULATE ABBREVIATED SOMETIMES ABBR ACCUM
Information entities can use the following naming conventions:

- Database elements can be established using approved names from the glossary and can be further defined with synonyms. Element names should have a maximum of 25 characters. The following example lists an element and three synonyms:

  EMPLOYEE-CODE

  DB-REC-EMPLOYEE-CODE
  MAP-EMPLOYEE-CODE
  WORK-EMPLOYEE-CODE

- Database Records can be composed of approved, usable names (for example, EMPLOYEE). Records can be given greater flexibility with the addition of suffixes. The following example lists employee records with identifying suffixes:

  EMPLOYEE-0600
  EMPLOYEE-2500
  EMPLOYEE-6359

In CA ADS process source, as well as in COBOL, CA Culprit, and map source, the elements can be referenced by the element name plus the suffix, as follows:

  EMPLOYEE-CODE-6359

- Map work records are composed of the map name followed by the suffix -MAP-RECORD, as in the following example:

  CEMMILIS-MAP-RECORD

  Elements in the map record utilize the prefix MAP- and the element name, as follows:

  MAP-OFFICE-CODE

  If the map needs more than one work record, a number is added to the word MAP, as follows:

  CEMMILIS-MAP2-RECORD (the second map record)
  MAP2-OFFICE-CODE (a record element from the second record)

- Dialog work records are composed of the dialog name followed by the suffix -WORK-RECORD as in the following example:

  CEMDULIS-WORK-RECORD

  Elements in the dialog work record utilize the prefix WORK- and the element name, as follows:

  WORK-OFFICE-CODE
If the dialog needs more than one work record, a number is added to the word WORK, as follows:

**CEMDILIS-WORK2-RECORD** (the second dialog work record)

**WORK2-OFFICE CODE** (a record element from the second record)

- Set names are established by concatenating an abbreviation of the owner record (a 7-character maximum) with that of the member record (a 6-character maximum), as follows:

  **EMPL-SKILL**

### Performance Considerations 1

The performance of the CA ADS runtime system is dependent upon a number of factors, such as the size of the CA IDMS/DC or DC/UCF system, the number of applications being run concurrently, and the number of users for a given application. Rather than attempting to give definitive instructions for the improvement of performance, this section discusses the following aspects of the CA ADS runtime system:

- Parameters affecting performance
- Resource management

Each of these considerations is discussed separately below.

### CA ADS System Generation

The CA ADS system is generated by submitting ADSO, PROGRAM, and TASK statements to the DC/UCF system generation compiler. Optionally, the KEYS statement is used to define site-specific control key functions.

- Statements (see page 166)
- Allocating Primary and Secondary Storage Pools (see page 168)
- Relocating Resources (see page 169)
- Specifying the Number of Online Tasks and External Request Units (see page 170)

#### Statements

These statements are used as follows:

- The ADSO statement includes parameters that define the CA ADS runtime environment, as follows:
  - The task code (ADS) that initiates the CA ADS runtime system
  - The mainline dialog that can begin executing immediately
  - The maximum number of dialog levels that can be established by each application
The disposition of record buffers during a pseudo-converse (that is, whether they can be written to the scratch area of the data dictionary)

The size of the primary and secondary record buffers

The AUTOSTATUS facility that handles errors generated by database, logical record, or queue and scratch record processing

The Status Definition Record that associates status codes returned by database, logical record, and queue and scratch record access with condition names

The treatment of numeric values placed into alphanumeric fields by arithmetic and assignment commands

The display of the Dialog Abort Information screen when the runtime system detects an abend condition in an executing dialog

Whether dialog statistics will be collected

How CA ADS is to perform a mapout when a dialog's map is already displayed as a result of a previous mapout in a pageable map

Whether record buffer blocks are to be compressed across a pseudo-converse when they are retained in the storage pool

How the amount of storage to be allocated for record buffer blocks is to be determined

The PROGRAM statement defines the following CA ADS components as DC/UCF programs:

The ADSORUN1, ADSORUN2, and ADSOMAIN runtime system programs

The system maps (the menu map, runtime message map, and maps for each of the application and dialog compiler screens)

The application and dialog compiler programs (ADSA and ADSC)

CA ADS dialogs (an optional parameter if null Program Definition Elements (PDEs) are defined in the SYSTEM statement)

The TASK statement defines the following task codes:

ADS and ADS2 to initiate the runtime system

ADSA to initiate the application compiler

ADSC to initiate the dialog compiler

ADSR to initiate the runtime system when returning from a linked user program

For detailed syntax and examples of the sysgen statements, refer to the CA IDMS System Generation Section.
The following discussion highlights selected aspects of system generation that have particular import when considering system performance in a CA ADS environment. These features are as follows:

- Allocating primary and secondary storage pools
- Relocating resources
- Specifying the number of online tasks and external request units

Each of these considerations is discussed separately below.

**Allocating Primary and Secondary Storage Pools**

**Record Buffer Block**

The runtime system allocates and initializes record buffers for use by executing dialogs. When an application is initiated, CA ADS allocates a Record Buffer Block (RBB) from the DC/UCF storage pool to hold the subschema, map, and work records accessed by the dialogs in the application thread. The RBB must be large enough to accommodate the largest of these records.

There is one primary RBB for each application. CA ADS allocates a secondary RBB when the RBB becomes full during execution of the application or does not have enough remaining space to hold a record. Additional secondary RBBs can be allocated by the CA ADS runtime system as necessary.

The data communications administrator (DCA) can specify the size of the primary and secondary RBBs with the PRIMARY POOL and SECONDARY POOL parameters of the ADSO statement. When allocating the primary and secondary storage pools, the DCA needs to consider the size and number of the records used by the application as well as the header records maintained by the buffers.

The primary RBB should be large enough to satisfy the records associated with the most-frequently used dialogs. The secondary RBB should be large enough to accommodate the largest record.

Alternatively, the runtime system can be directed to calculate the size of the RBBs for an application or dialog and to use the calculated size when acquiring storage space for the RBBs. Use the calculated size in systems where there is a high number of records and storage space is a concern.

The following diagram shows the structure of the Record Buffer Block. Each record buffer contains a 24-byte header to keep track of available space. For each record in the pool, CA ADS maintains a record header (RBE) that requires 44 bytes of storage for database records and 56 bytes of storage for logical records. There is also a header for each element of a logical record. All records and headers are aligned on doubleword boundaries. Each buffer pool must be large enough to accommodate the largest subschema, map, work, database, or logical record used by a dialog in the application.
Relocating Resources

The fast mode threshold is used by the CA ADS runtime system in conjunction with the RESOURCES ARE FIXED specification to determine whether record buffers are written to disk or kept in main storage across a pseudo-converse. If the total size of all record buffers, in bytes, exceeds the fast mode threshold and RESOURCES ARE FIXED has been specified, the record buffers are written to disk; otherwise, the record buffers are kept in the storage pool. Storage used for currency blocks, CA ADS terminal blocks (OTBs), OTB extensions, and variable dialog blocks (VDBs) is not eligible for writing to scratch.

Size of the Threshold

The size of the threshold is a site-specific determination that is based on the availability of general resources versus the amount of available storage. I/Os for DC/UCF journaling and CPU cycles for record locking are used when record buffers are written to the scratch/queue areas. Therefore, when buffers exceed the fast mode threshold, the increased use of resources will slow down the transaction response time. On the other hand, if buffers are always under the threshold (that is, if the fast mode threshold is high), more memory is required.

Alternative Method
Alternatively, the DCA can specify that storage used for RBBs and statistics control blocks is always written to scratch across a pseudo-converse, regardless of the relocatable threshold that has been defined for primary and secondary storage pools. In this situation, other storage (that used for currency blocks, OTBs, OTB extensions, and VDBs) is written to scratch across a pseudo-converse only when the relocatable threshold is exceeded.

Specifying the Number of Online Tasks and External Request Units

The MAXIMUM TASKS and MAXIMUM ERUS parameters specify the maximum number of user tasks (online tasks) and external request units that can be active concurrently. The size of these parameters can affect the amount of time spent by DC/UCF in searching the queues for tasks that are waiting to be executed.

Considerations

The number of online tasks and external request units that should be specified is a site-specific determination and is dependent upon factors such as the number of tasks processed each hour in a particular environment. When setting the MAXIMUM TASKS and MAXIMUM ERUS parameters on the SYSTEM statement, the following statistics should be considered:

- Increasing the MAXIMUM TASKS or MAXIMUM ERUS parameters by one (1) causes virtual storage requirements to increase as shown below:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Size of resource</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCE</td>
<td>248 bytes</td>
<td>248 bytes</td>
</tr>
<tr>
<td>STACKSIZE</td>
<td>320 words</td>
<td>1,280 bytes</td>
</tr>
<tr>
<td>DCE</td>
<td>40 bytes</td>
<td>40 bytes</td>
</tr>
<tr>
<td>ECB * 3</td>
<td>8 bytes</td>
<td>24 bytes</td>
</tr>
<tr>
<td>DPE * 20</td>
<td>16 bytes</td>
<td>320 bytes</td>
</tr>
<tr>
<td>RCE * 15</td>
<td>16 bytes</td>
<td>240 bytes</td>
</tr>
<tr>
<td>RLE * 25</td>
<td>8 bytes</td>
<td>200 bytes</td>
</tr>
<tr>
<td>Total increase per task</td>
<td>2,352 bytes</td>
<td></td>
</tr>
</tbody>
</table>

Note that a value larger than the default (420) should be specified for the STACKSIZE when using CA ADS. If the STACKSIZE is at 420 and two tasks exceed stacksize and go into abend storage at the same time, the system will abort with an abend code of 3995.

- The following DC/UCF system parameters should be increased as specified for every increment of 1 in the size of MAXIMUM TASKS or MAXIMUM ERUS:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Amount increased</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB LIST</td>
<td>3</td>
</tr>
<tr>
<td>DPE COUNT</td>
<td>20</td>
</tr>
<tr>
<td>RCE COUNT</td>
<td>15</td>
</tr>
<tr>
<td>RLE COUNT</td>
<td>25</td>
</tr>
</tbody>
</table>
ADS Resource Management

In designing applications, consideration must be given to the efficient management of system resources. The management of resources such as the database, the storage pool, and the program pool storage affects the performance of online applications since many users may require access to these resources simultaneously.

- Monitoring Tools (see page 172)
- Task Processing Support (see page 172)
- Variable Storage Pool (see page 173)
- Program Pool Storage (see page 174)
- Database Locks (see page 174)
- Disk I/O (see page 175)
- Terminal I/O (see page 175)
- CPU Usage (see page 176)
- Conserving Resources (see page 176)

The following diagram shows the resources used by an application in a non-SQL environment while a task is active and after the task has terminated.

**When a Task is Active**

<table>
<thead>
<tr>
<th>Control blocks</th>
<th>Logical Terminal Element (LTE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Control Element (TCE)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program pool</th>
<th>Storage pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Dialog Block (ADB)</td>
<td>Online Terminal Block (OTB)</td>
</tr>
<tr>
<td>Fixed portion of subschema (IB50)</td>
<td>Online Terminal Block Extension (OTBX)</td>
</tr>
<tr>
<td>Fixed Dialog Block (FDB)</td>
<td>Variable Dialog Block (VDB)</td>
</tr>
<tr>
<td>Map (MCH)</td>
<td>Record Buffer Block (RBB)</td>
</tr>
<tr>
<td>Online Work Area (OWA)</td>
<td>Currency Block</td>
</tr>
<tr>
<td>Requestor Locking Table (RLT)</td>
<td>Variable Subschema Block (VB50)</td>
</tr>
</tbody>
</table>

**After the Task has Terminated**
The remainder of this section discusses methods that can be used to monitor the resource consumption of an application and ways in which to utilize available resources efficiently.

Monitoring Tools

As with any task running under DC/UCF, the major resources to be monitored in a CA ADS environment are as follows:

- Task processing support
- Variable storage pool
- Program pool storage
- Database locks
- I/Os (disk and terminal data transmission)
- CPU cycles

Each of these resources can be monitored with data dictionary reports and DC/UCF master terminal functions, as discussed below.

Task Processing Support

The following diagram shows the resources in use while a task is active (left diagram) and those in use after the task (right diagram) terminates.
### While a task is active:

- TCE
  - RLE
  - RCE
  - Resource

### After a task has terminated:

- LTE
  - RLE
  - RCE
  - Resource

#### IDMSDB--Task Processing Support

### Displaying Internal Resources

The following DC/UCF master terminal functions display the internal resources used to support task processing:

- **DCMT DISPLAY ACTIVE TASK** displays global statistics on active tasks and information on each active task thread.

- **DCMT DISPLAY STATISTICS SYSTEM** displays information on the system including the peak task control element (TCE) stack; and the maximum number of resource link elements (RLEs), resource control elements (RCEs), and deadlock prevention elements (DPEs) used by the tasks.

#### Variable Storage Pool

The following sysgen reports (CREPORTS) and DCMT functions can be used to monitor the use of the storage pool:
- CREPORT 25 verifies the size of the storage pool and indicates whether storage protection has been enabled for the system.
- DCMT DISPLAY ACTIVE STORAGE shows the current fragmentation of the storage pool.
- DCMT DISPLAY LTERM RESOURCES indicates which terminals are active and own resources.
- DCMT DISPLAY LTERM $lterm-id-a RESOURCES displays the specific resources (and the addresses of those resources) owned by the named terminal.
- DCMT DISPLAY MEMORY $hex-address-a displays an actual resource as it appears in memory.
- CREPORT 40 supplies the current parameters specified in the ADSO statement.

The CA IDMS Reports Section section describes CREPORTS; the CA IDMS System Tasks and Operator Commands Section details the master terminal functions available to monitor system resources.

Information from the above displays and reports can be used to calculate the number of users the system can currently support, assuming various storage pool sizes.

Program Pool Storage

The following DCMT commands can be used to provide information on the program pool:

- DCMT DISPLAY ACTIVE PROGRAMS displays the following:
  - Statistics on program pool usage, including the total number of pages and total number of bytes in the pool; the number of loads to the program pool; the number of pages loaded; and the number of load conflicts
  - Information on currently active programs including the program name, type, and version number; count of users currently using the programs; size of the program in K bytes; the number of times the program was called; and the number of times the program was loaded into the program pool
  - The program pool page allocation map that shows which pages are not in use; which pages are in use by one program; and which pages are in used by more than one program
- DCMT DISPLAY ACTIVE REENTRANT PROGRAMS displays the above information for the reentrant program pool and the active reentrant programs. If no reentrant pool is defined, the standard program pool is shown.

Database Locks

The DCMT DISPLAY RUN UNIT and OPER WATCH DB RUN UNITS commands can be used to show the number of database locks being requested for a particular run unit. The number of database locks maintained by a DC/DC system has considerable impact on CPU usage.

These locks are specified at sysgen time by the RULOCKS and SYSLOCKS parameters of the SYSTEM statement.
For a discussion of database locks, refer to the CA IDMS Database Design Section. For information on factors to consider when preparing the SYSTEM statement, refer to the CA IDMS System Generation Section.

Disk I/O

The following reports can be used for monitoring disk I/O:

- JREPORT 004 shows the average number of I/Os to disk for a given program.
- DCMT DISPLAY RUN UNITS or OPER W DB RU shows if any run units are waiting for a journal buffer (as indicated by a run unit status value of IUH). IUHs occur most frequently when the fast mode threshold is set too low.

For information on the JREPORTS (journal reports), refer to the CA IDMS Reports Section.

Terminal I/O

Monitoring Steps

The following steps can be taken to monitor terminal I/Os:

1. Run the mapping utility (RHDCMPUT) for a report on a specific map. This report will display a picture of the map and the attributes currently assigned to the map. The report will also indicate whether BACKSCAN is enabled for any mapping fields. If BACKSCAN is in effect and the NEWPAGE option on the ADSO statement has been selected, extraneous data from the previous mapout may be left on the screen when a map is redisplayed. It is advantageous to have NEWPAGE in effect, however, because this option increases runtime efficiency by reducing the number of data fields that need to be transmitted to the terminal. For more information on the NEWPAGE feature, refer to the CA ADS Reference Section.

2. Use DCMT VARY PTERM pterm-id TRACE ALLIO FF to cause the datastream being transmitted to the terminal to be written to the log as well.

3. Use SHOWMAP map-name in conjunction with DCUF USERTRACE to cause the datastream of a particular map to be traced.

4. Use DCMT VARY PTERM pterm-id TRACE ALLIO OFF to turn off the trace, suppressing any further transmission of datastreams to the log.

5. Run the print log utility (RHDCPRLG) to show the actual trace. Specify the following parameters in the utility JCL:

   PRINT ALL FOR ALL
   FROM time ON date
   TO time ON date

   Transmission times can be calculated by analyzing the length of the datastream.
CPU Usage

To monitor CPU cycles and obtain CPU usage by task, the system can be instructed to collect task statistics. It is advisable not to request task statistics unless there is a demonstrated need as they require considerable overhead and generate a large volume of data. Task statistics are requested by specifying TASK STATISTICS WRITE or TASK STATISTICS COLLECT on the SYSTEM statement. The statistics are written to the DC/UCF log.

For detailed information on collecting task statistics, refer to CA IDMS System Operations Section.

Conserving Resources

Resources can be conserved as follows:

- Enable storage protection -- Storage protection is enabled by specifying PROTECT in the SYSTEM statement at system generation. The benefits of using storage protection are as follows:
  - CPU overhead is reduced because there are shorter chains for the system to walk.
  - Resources are clustered.

To avoid SVC overhead, it is advisable to enable storage protection (that is, specify PROTECT) on the SYSTEM statement and to disable storage protection (that is, specify NOPROTECT) on the PROGRAM statement.

- Specify buffer sizes in multiples of 4084 bytes -- The 4084-byte limit represents a multiple of 4K (4096 bytes) less the 12 bytes for pointer information and task ID address, as shown below:

<table>
<thead>
<tr>
<th>RCE access</th>
<th>Task ID</th>
<th>Actual storage</th>
<th>RCE address</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bytes</td>
<td>4 bytes</td>
<td>4084 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

If a 4K page were selected, storage would have to be taken from two contiguous pages. The benefits of placing a 4084-byte limit on the amount of storage acquired are as follows:

- Fragmentation of the storage pool is reduced when only one page is requested. Space is allocated in contiguous frames for a particular request. It is easier for the system to find one page rather than two contiguous pages.

- Less CPU overhead is required because partial pages do not have to be calculated or scanned. When the system finds a request for a multiple of 4K (less the pointer information), it will immediately scan the pool looking for entirely empty pages, thus saving overhead.

- Limit the size of subschemas -- Subschemas should be specified to the requirements of the application. The size of the currency block is directly related to the storage requirements of the variable subschema storage block (VB50) used at runtime; the runtime system maintains currency tables for every record, set, and area accessed by the dialog. Therefore, it is worthwhile to make subschemas as streamlined as possible.
Limit the number of dialog levels -- The MAXIMUM LINKS parameter of the ADSO sysgen statement specifies the maximum number of dialog levels that can be established by each respective CA ADS application; keep this parameter low. A well designed application has as few levels as possible. The number of levels should be limited because, for each level established in the application, kept storage is acquired for the Variable Dialog Block (VDB) and the currency block. Storage established at a particular level is not released until control is passed upward. To limit the number of levels established, use the TRANSFER command whenever possible; build the application horizontally (that is, pass control laterally) rather than vertically.

Control the size of the application -- The size of dialog premap and response processes, the number of data fields included in a map, the size of the subschemas, and the size of database, map, and work records affect the performance of the CA ADS runtime system. The actual number of I/Os required to load a complete program is dependent upon the size of a page in the DDLDCLOD area, the amount of overflow that will be encountered to load that record, and the size of the actual program being loaded. Therefore, the following benefits are realized by minimizing the size of programs:

- A reduction in the work required to load a small program as compared to a large program
- A reduction in time spent loading a particular program in the program pool or reentrant pool
- A reduction in time spent waiting for space in the program pool or reentrant pool

Under DC/UCF, the term program refers to CA ADS dialogs, tables, maps, subschemas, and online and batch programs.

Make frequently called programs resident -- A frequently called program (such as ADSOMAIN) is virtually a resident in the program pool or the reentrant pool. The program should be made resident because the operating system can page more rapidly than IDMS-DC/UCF can read in a page from the DDLDCLOD area. By making the program resident, the operating system, rather than DC/UCF, will be requested to bring the page in core. Additionally, the program and resident pool will be less fragmented when a frequently used program is made resident. A program can be specified as resident on the PROGRAM statement at system generation.

Free the resources of an inactive terminal -- The resource timeout facility can be activated on the SYSTEM statement at system generation, specifying the amount of time a terminal is permitted to be inactive (that is, have no task executing) before all resources owned by the terminal are deleted and control is returned to the system. Because longterm storage resources are associated with a terminal even though a program is not active, freeing those resources will free space for other users of the system. This is particularly important if longterm locks are being implemented.

Overview of CA ADS Application Development

CA' Application Development System (CA ADS) enables you, as an application developer, to develop and execute online applications that query and update a CA IDMS/DB database or VSAM file.

Development Tools

The CA ADS application development environment features menu-driven development tools that simplify and speed the definition of applications. Components developed by using CA ADS development tools are modular and can be reused in one or more applications.
Process Language

To enable an application to access the database, perform calculations, and perform other customized processing, you use the high-level process language provided by CA ADS. Easily recognizable high-level process language commands (such as DISPLAY) greatly simplify many frequently performed operations. Modules written in this process language can be added to an application whenever processing is required.

CA ADS is fully integrated with CA’ Application Development System/Batch (CA ADS Batch). Developers who use CA ADS Batch to define batch applications can use the same development tools used by CA ADS developers.

For more information on CA ADS Batch, see the CA ADS Batch User Section.

Additionally, CA ADS applications can accommodate components developed by using the CA IDMS Automatic System Facility (ASF). Execution of CA ADS applications can be traced and debugged by using the CA IDMS online debugger or CA ADS Alive. Runtime performance and resource usage of an application can be tracked by using the CA IDMS Performance Monitor.

This section introduces CA ADS by discussing:

- Application development in the CA ADS environment
- Application development tools that will be discussed in this user’s section

Application Development

Flexible Application Development Environment

CA ADS provides a flexible application development environment. You can define application components in whatever order makes the most sense given the application you need to define. For example, you can define the executable structure, screen displays, and runtime flow of control for the entire application before you define any modules of process code for the application. Alternatively, you can fully define all components, including modules of process code, for a subset of the application before even beginning to define components for the remaining application.

Methodology

This section illustrates CA ADS concepts by showing you how they apply to the definition of a sample application. The methodology presented here shows you one way to efficiently develop a new application by using CA ADS development tools. At your site, other application development strategies may be employed.

Structure Diagram

A useful first stage in developing a new CA ADS application is to develop a structure diagram based on the specifications for the application. Drawing a structure diagram for an application is useful because:

- The application’s functional requirements are clarified before any specific development occurs. Potential design and development misunderstandings are often caught in the process of developing a structure diagram.
The structure diagram is a development resource for application developers throughout the development cycle. While developing application components, the developer can use the structure diagram as a reminder of how each component relates to the entire application. CA ADS application components fit directly into an application structure diagram.

Sample Application

The structure diagram for a small Personnel application is shown below. This application allows users to maintain information on departments and users. Related data, such as office addresses and insurance information, is also collected and stored by using this application. You will develop the Department portion of this application in this section.

IDMSDB--Application Development

Functions

As shown in this sample structure diagram, the application structure is made up of functions. Each function is related to work performed by the end user on a single screen. For example, the Personnel application includes functions such as ADDDEP, MODDEP, and DELDEP, which allow you to add, modify, and delete department records in the database. Menu functions, such as MAINMENU and DEPTMENU, allow you to select other functions in the application. Other functions allow you to return to previous menus (functions BACK and TOP) and to exit from the application (function EXIT).

To reduce repetition of information, this section presents steps for developing only the Department portion of this sample Personnel application. Remaining portions of the Personnel application can be added to the Department application at any time. The strategy of defining different portions of applications at different times is particularly useful when developing large applications.

Defining the Application
As soon as the application structure diagram is completed and approved, you can begin to define the application in the data dictionary. In this section, you will define the sample Department application in two steps:

1. **You will develop a prototype application** based on the approved structure diagram for the application. The CA ADS runtime system simplifies the definition of prototypes by automatically performing many basic processing activities, such as displaying a screen. You can define a prototype without writing any process logic. Additionally, you can execute a prototype application before the database is developed.

2. **You will enhance the tested and approved prototype** by adding process logic.

**Advantages**

Advantages to developing a prototype application before defining any process logic for the application include:

- **Timely feedback** -- End users and other interested people can execute the completed prototype application to see how well screens meet their needs and to verify that flow of control from function to function makes sense based on their job responsibilities.

- **Ease of modification** -- Improvements suggested by end users can be included in the prototype application quickly because you do not have to modify any process logic or database definitions.

- **Efficient use of development time** -- The prototype application developed for early testing is itself developed into the final production application. Components created for the prototype are all used in the final application.

**Application Development Tools**

CA ADS applications are developed and executed by using a variety of online tools. The following tools are used to develop the sample Department application.

- **CA ADS application compiler (ADSA)** -- Used to define the executable structure of an application, based on the structure diagram developed for the application.

- **CA ADS dialog compiler (ADSC)** -- Used to define CA ADS dialogs, which handle most runtime interactions with the end user.

- **Online mapping facility (MAPC)** -- Used to define maps, which establish preformatted screens. Dialogs use maps to display and allow end users to input information.

- **Integrated Data Dictionary (IDD) menu facility** -- Used to create data definitions and modules of process code.

- **Runtime system** -- Used to execute CA ADS applications at any stage in the application’s life cycle.
IDMSDB--Application Development Tools

As an application developer, you can execute the application at any time in the development cycle by using the runtime system. Additionally, end users execute the application by using the runtime system.

Components defined by using ADSA, ADSC, MAPC, and the IDD menu facility are all stored in the data dictionary.

You can access any of the above development tools from CA IDMS/DC, the CA IDMS teleprocessing (TP) monitor, or DC/UCF, the teleprocessing monitor interface. Additionally, you can transfer directly among ADSA, ADSC, MAPC, and the IDD menu facility by using the transfer control facility (TCF).

Transfer Control Facility You can use TCF at any time during an application development session to suspend one development tool and transfer to another.

For example, while using ADSC to define a dialog, you might remember that the related map definition is still incomplete. You can suspend your ADSC session, transfer directly to MAPC to complete the map, and then transfer back to ADSC to resume your suspended dialog-definition session.

Task Codes

To invoke a development tool from CA IDMS/DC or DC/UCF (DC/UCF), or TCF, specify the task code associated with the tool. A task code is a unique invocation name defined for a development tool at system generation time. Sample task codes for CA ADS development tools are presented in the table below. The task codes shown allow you to operate under TCF and, therefore, switch from tool to tool.

⚠️ Note: If you are not operating under TCF, you cannot switch to another tool without first returning to DC/UCF.
Task codes can vary from site to site.

<table>
<thead>
<tr>
<th>Development tool</th>
<th>Sample task code</th>
<th>Site task code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA ADS application compiler (ADSA)</td>
<td>ADSAT</td>
<td></td>
</tr>
<tr>
<td>CA ADS dialog compiler (ADSC)</td>
<td>ADSCT</td>
<td></td>
</tr>
<tr>
<td>IDD menu facility</td>
<td>IDDMT</td>
<td></td>
</tr>
<tr>
<td>Online mapping (MAPC)</td>
<td>MAPCT</td>
<td></td>
</tr>
</tbody>
</table>

### Defining an Application Structure Using ADSA

As an application developer, you can begin defining application components as soon as the application structure diagram is complete. You can use CA ADS development tools to define a preliminary, or **prototype** version of the application early in the application development cycle. Application developers and users can execute the prototype to test and suggest improvements to the application.

As soon as the prototype application is approved, definitions that make up the application can be enhanced to perform all processing required by the final production application. In this part of the section, "Developing the Prototype," you will develop a prototype Department application. Then you will take this prototype application and develop it into a fully functional application.

The first step in defining a prototype application in the data dictionary is to define the application's structure, based on the structure diagram, by using the CA ADS application compiler (ADSA). The structure of an application specifies how a user moves from function to function when executing the application.

In this section, you will begin defining the sample Department application introduced in Section 7, "Overview of CA ADS Application Development". This section includes:

- An overview of developing an application structure in the CA ADS environment
- Instructions for defining the sample Department application
- A summary of what you've accomplished in this section

### Functions

An application is made up of the various **functions** necessary to do the work of the application. Each function represents a unit of work in the application. For example, different functions in the Department application:

- Display a menu to help the user navigate through the application
- Perform functional processing, such as allowing the user to add or modify a department record
- Perform standard system activities, such as returning control to the application's main menu or exiting the user from the application
Responses

The development team must arrange application functions so that users can easily get from one function to another. The team defines paths that a user can take between functions. Each path is called a response. At runtime, the user can select from among available responses to get from one function to another.

Application structure

The arrangement of functions and responses constitutes the structure of a CA ADS application. The diagram below shows the structure of the sample Department application. This diagram includes both functions and responses because you need to define both types of application components. The structure diagram for an application serves as a blueprint for application development. Functions and responses in this sample application will allow users to add, modify, and delete information on departments.

⚠️ Note: In this sample application, you can substitute your initials for the XXX in the task code, application name, dialog names, etc.
Types of functions

Most functions in a CA ADS application, including the Department application, are menu, dialog, or system functions.

Menu function

A menu function displays a menu screen to an end user.

For example, the DEPTMENU function in the Department application is a menu function. The menu screen displayed by DEPTMENU lists the ADD, MOD, DEL, and EXIT responses. At runtime, the user can select any of these responses from the DEPTMENU menu screen. To define a menu function, all you have to do is specify where the menu belongs in the application structure. CA ADS provides predefined screen formats and processing logic for menu functions.

Dialog function
A dialog function typically displays a screen that supplies information to or requests information from a user. Based on entries made by the user, the function performs processing activities such as data retrieval and update. As a developer, you define CA ADS programs, called dialogs, to handle runtime operations for dialog functions.

In the sample Department application, functions ADDDEP, MODDEP, and DELDEP are dialog functions. The user can use ADDDEP to add a new department record, MODDEP to modify an existing department record, and DELDEP to delete a department record from the database.

System function

A system function displays a screen and/or performs operations that are common to most applications.

For example, the POP system function in the sample Department application returns the user to the most recently used menu screen (in this case, the DEPTMENU menu). The QUIT system function allows the user to exit from the application at any point. To define a system function, all you have to do is specify where the function belongs in the application structure. CA ADS provides all necessary logic and screens for system functions.

Paths between functions

Responses define paths between functions. At runtime, a user moves from one function to another by selecting a response that leads from the first function to the next. For example, a user of the Department application can get from DEPTMENU to ADDDEP by entering a nonblank in the selection field to the left of the ADD response on the menu.

The user selects a response either by typing its name (for example, ADD) in a special response field on the screen or by pressing the function key (for example, [PF4]) associated with the response. Default responses can be defined for functions, making it easier for users to get to frequently used functions. The structure diagram shown above gives the name and function key for each response in the sample Department application.

Worksheets

A typical application has numerous responses and functions. Some responses and functions may be available in several places in the application. To help keep track of the responses and functions in an application structure, some sites develop a worksheet of responses and functions for the application. A standard worksheet that documents each response and function is especially useful when you are defining the application structure in the data dictionary. Filling out a worksheet for application responses and functions can simplify the application development process and can also provide documentation of application development decisions.

A sample worksheet for the Department application is shown below.
Task codes

To enable users to access an application, you define entry points into the application structure. You establish entry points for CA ADS applications by defining task codes. When you define a task code, you associate it with a specific application function. At runtime, you use the task code to begin executing the application at the associated function. Both application developers and users invoke applications by using task codes.

For example, the DEPT task code is associated with the DEPTMENU function in the sample Department application shown previously. A user can invoke the application from a CA IDMS/DC or DC/UCF system by specifying DEPT and pressing [Enter]. The DEPTMENU menu screen is the first application screen shown to the user.

Multiple task codes

You can define more than one task code for an application. Defining multiple task codes, or entry points, is particularly useful for very large applications. For example, if the sample Department application is incorporated into a large, company-wide application, users who only need to use department information can use the DEPT task code to enter the application at the DEPTMENU function. Clerks in the Accounts Receivable office can use a different task code to enter the company-wide application at a menu that lists billing and customer functions.

Note: In the sample application, you can substitute your initials for the XXX in the task code.

Instructions

As a member of the development team, you use the CA ADS application compiler (ADSA) to define an executable structure for an application. ADSA screens lead you through the steps necessary to define the functions, responses, and task codes that make up the application’s structure.
ADSA screens

ADSA screens are designed in accordance with the CUA definition of SAA. All screens have:

- A title at the top
- A data entry area
- PF keys at the bottom

Menu screens also have an action bar, a message area, and a command line.

```
- Add Modify Compile Delete Display Switch
- CA ADS Application Compiler
- CA, Inc.
```

```
Application name . . . . _______
Application version . . ____
Dictionary name . . . . -------
Dictionary node . . . . -------
Screen . . . . . . . . . . 1. General options
                      2. Responses and Functions
                      3. Global records
                      4. Task codes
```

```
Copyright (C) 1999 CA, Inc.
Command ===>
Enter F1=Help  F3=Exit  F10=Action
```

Steps

To define the sample Department application structure, you will perform the following steps:
1. Invoke ADSA
2. Name the application
3. Specify basic information on the application
4. Define application response and function relationships
5. Further define the application responses
6. Further define the application functions
7. Define the application task code
8. Compile the application (creating an application load module)

After you finish defining the Department application structure, you can exit from ADSA and optionally execute the application to test out the definitions that you have made. If you need additional information at any time about the use of ADSA, see "Using ADSA" in section B.

⚠️ **Note:** Only the menu access can be tested until prototype maps and dialogs are created.

The following diagram shows an overview of the flow and structure of ADSA.

⚠️ **Note:** This overall structure applies to the MAPC and ADSC compilers as well as to ADSA.
Step 1 Invoke ADSA

To invoke ADSA, you must be signed on to a DC/UCF system. Procedures for signing on to DC/UCF differ from site to site. If you are not sure how to sign on, ask other users at your site for help.

Before you invoke ADSA, you may want to establish a default dictionary and/or Distributed Database (DDS) node for the current definition session under DC/UCF. Naming a default dictionary is useful if you are working in a multiple-dictionary environment. Naming a DDS node is useful only if you are working in a distributed database network.

At some sites, signon profiles for each user establish the appropriate default dictionary and/or node for each user that signs on to CA IDMS/DC or DC/UCF.

If you want to establish defaults for your current DC/UCF session, enter the appropriate commands, one at a time, after signing on to DC/UCF:

- You name a default dictionary by entering:
  
  `DCUF SET DICTNAME dictionary-name`

- You name a default DDS node by entering:
  
  `DCUF SET DICTNODE node-name`

You can display the names of your default dictionary and DDS node at any time by entering, one at a time:

  `DCUF SHOW DICTNAME`

  `DCUF SHOW DICTNODE`

**Specifying the task code for ADSA**

When you’ve signed on to DC/UCF and optionally established a default dictionary name/node, you can invoke ADSA. To do this, you enter the task code for ADSA1.

For example, if the task code for ADSA is ADSAT, you can invoke ADSA from CA IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
adsat
```

Press the ENTER key to input the -- [Enter] task code for ADSA.

Using the task code ADSAT means that you are invoking ADSA under TCF (the transfer control facility). Once you are in the ADSA tool under TCF, you can switch to another application development tool without returning to DC/UCF.

**Main menu screen**

ADSA begins with the Main Menu screen:

```
Add  Modify  Compile  Delete  Display  Switch
```

CA ADS Application Compiler
Step 2 Name the Application

The ADSA Main Menu screen allows you to specify the name of the application, its version number, and the dictionary in which it resides. This dictionary name overrides the current default (if one has been set) and becomes the new default dictionary.

Screen prompts

When you begin an application definition, you typically enter information after one or more of the following Main Menu screen prompts:

- **Application name** -- You must supply an application name. The name you specify must be unique among all program names (including applications, dialogs, maps, tables, help modules, etc.)

- **Application version** -- You can optionally type a version number for the application after this prompt. When following the example in this section, use the default version number of 1. In a development environment, you might select a different test version number, such as 5. You would give all test components the same test version number. You would then set up the system so that, when you execute the application, test components are executed rather than production components.
  For more information on maintaining separate test and production definitions in the same data dictionary, see *CA IDMS System Operations Section*

- **Dictionary name** -- A dictionary name may already be displayed after the Dictionary name prompt on your terminal. A sample dictionary name (DEMO) is shown in examples throughout this section.
  If a dictionary name is not displayed, check with others at your site to find out if you need to specify one. You typically need to enter the name of your group’s dictionary only when you are not using the default dictionary in a multiple-dictionary environment.
Dictionary node -- A dictionary node name may already be displayed after the Dictionary node prompt on your terminal. Sample definition screens in this section do not show a node name. If a dictionary node is not displayed, check with others at your site to find out if you need to specify one.

You can use the tab key to move the cursor quickly and easily between prompts. Begin defining your application on the Main Menu screen as shown below:

Add Modify Compile Delete Display Switch

CA ADS Application Compiler

CA, Inc.

Application name . . . x x x a p p l _
Application version . . . 1
Dictionary name . . . demo____
Dictionary node . . . _______

Adding the application

To add the application, position the cursor on the Add item on the action bar and press [Enter]. You can position the cursor on Add by:

• Pressing [PF10] move to the action bar and then tabbing to Add and pressing [Enter]
• Tabbing to Add and pressing [Enter]
• Typing add on the command line and pressing [Enter]

Add Modify Compile Delete Display Switch

Copy from applicationA-ADS Application Compiler
Name

Application name . . . XXXAPPL_
Application version . . . 1
Dictionary name . . . DEMO____
Dictionary node . . . _______

Once you have displayed the Add action item, press [Enter] to add the application to the dictionary. The action is confirmed.

ADSA redisplays the Main Menu screen with an appropriate message:

• A confirming message is returned when your definition contains no errors.
• An error message is returned if ADSA detects any errors. Read the message to see what problem has occurred. You can type over any fields in error and then press [Enter] again.
Step 3 Specify Basic Information

After you specify the name of the application, you can give some basic information on the Department application on the General Options screen. You reach the General Options screen by entering 1 next to Screen on the Main Menu.

```
Add  Modify  Compile  Delete  Display  Switch
```

CA ADS Application Compiler
CA, Inc.

Application name . . . . xxxappl_
Application version . . 1
Dictionary name . . . . demo___
Dictionary node . . . . _______

Screen . . . . . . . . 1
1. General options
2. Responses and Functions
3. Global records
4. Task codes

Copyright (C) 1999 CA, Inc.

Command ==>
Enter F1=Help  F3=Exit  F10=Action

The General Options screen is displayed.

The General Options screen

```
General Options
Page 1 of 2
```

Application name: XXXAPPL  Version: 1

Description . .

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

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

Execution environment . . . . . . . . . . . 1
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

Screen prompts

You typically enter a description on the General Options screen:

- **Description** -- You can optionally type a one-line description of your application after this prompt.
After you specify basic information on the Department application, you can define the application's responses and functions as described below. Pressing [PF5] from the General Options screen will bring you to the Response/Function List screen.

**Step 4 Define Application Response and Function Relationships**

The **Response/Function List** screen is the main ADSA screen. You can reach this screen by pressing [PF5] from the General Options screen or by entering **2** at the **Screen** prompt on the Main Menu.

**Sample screen**

<table>
<thead>
<tr>
<th>Select (/)</th>
<th>Response name</th>
<th>Assigned key</th>
<th>Select (/)</th>
<th>Function name/type(1,2,3)*</th>
<th>Program/Dialog name</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>____________</td>
<td>_____</td>
<td>_</td>
<td>__________/</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>____________</td>
<td>_____</td>
<td>_</td>
<td>__________/</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>____________</td>
<td>_____</td>
<td>_</td>
<td>__________/</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>____________</td>
<td>_____</td>
<td>_</td>
<td>__________/</td>
<td></td>
</tr>
</tbody>
</table>

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

Each response in an application defines a path from one function to another. You are going to name the responses and functions and identify the key associated with each response and the dialog (if any) associated with each function as well as the function type.
Department application responses and functions

Below are the responses and functions invoked in the sample Department application. Each response is associated with a control key and invokes a function. A function can be associated with a dialog.

<table>
<thead>
<tr>
<th>Response name</th>
<th>Key</th>
<th>Function name</th>
<th>Function type</th>
<th>Dialog name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACK</td>
<td>CLE POP AR</td>
<td>Takes the user from ADDDEP, MODDEP, or DELDEP back to the DEPTMENU function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>PF9 QUIT</td>
<td>Exits the user from the application; EXIT is available from any function in the Department application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>PF4 ADDDEP Dialog XXXDA DD</td>
<td>Takes the user from the DEPTMENU function to the ADDDEP function, which allows the user to add a new department record.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOD</td>
<td>PF5 MODDEP Dialog XXXDU PD</td>
<td>Takes the user from DEPTMENU to the MODDEP function, which allows the user to modify an existing department record.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL</td>
<td>PF6 DELDEP Dialog XXXDU PD</td>
<td>Takes the user from DEPTMENU to the DELDEP function, which allows the user to delete an existing department record.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEPTME NU Menu</td>
<td>Provides a menu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entering Department application responses and functions

You name each response and function on the Response/Function List screen as shown below. Responses and dialog, program, and menu functions need further definition on subsequent screens. Select each response and function requiring further definition by entering a nonblank (nonunderscore) character under Select as you go along.

⚠️ Note: System functions, such as pop and quit are defined by CA ADS and do not require further definition on the part of the developer.
* Type: 1. Dialog  2. Program  3. Menu

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

Responses

For each response, enter the response name and name the assigned key.

Functions

For each function, enter the function name and specify the type (1 - dialog; 2 - program; 3 - menu) and the dialog associated with it, if there is one.

⚠️ Note: The dialog names can be specified early in the prototyping process; the dialogs need not exist.

Unique identifier for the response

The combination of the name, key, and associated function for a response makes up the unique identifier for the response within the application definition. Because of this, you can have several different responses with the same name and/or control key. For example, several ADD responses can be defined for an application. One ADD response can invoke function ADDDEP. Another ADD response can invoke function ADDEMP. This capability allows you to define a consistent user interface for an application, where all similar functions are invoked in similar ways.

Once you have named all responses and functions, you are ready to further define each one. Pressing [PF5] will advance you through the responses and functions selected on the Response/Function List screen.

Below is a discussion of defining responses and functions.

Step 5 Further Define the Application Responses

You further define each application response by using the Response Definition screen. A sample Response Definition screen is shown below:

Sample screen

Response Definition

| Application name: | XXXAPPL | Version: | 1 |
| Response name:    | ADD     | Function invoked: | ADDDEP |
| Description . . . | ____________________________ | Security class: | 0 |
| Response type. . . | 2. Global 2. Local |
| Response execution | 2. Immediate 2. Deferred |
| Assigned key . . . | PF04 |
Screen prompts

Certain information will be presented to you when this screen is displayed:

- **Application name** -- The name of the application will be displayed.
- **Version** -- The version number of the application will be displayed.
- **Response name** -- The name of the response as listed on the Response/Function List screen will be displayed. At runtime, the user can enter the response name on the screen to select the response.
- **Function invoked** -- The name of the function as listed on the Response/function List screen will be displayed. This is the function that the response invokes at runtime. For example, the ADD response invokes the ADDDEP function at runtime.
- **Assigned key** -- The key assigned to this response on the Response/Function List screen will be displayed. At runtime, the user can press the specified control key to select the response.

When you define a response on the Response Definition screen, you typically enter information after the following prompts:

- **Description** -- You can type a one-line description for each response. At runtime, the description is displayed on each menu from which the response is available.
- **Security class** -- You can assign a security class to this response. Applicable to online applications only, security class specifies the DC/UCF security class in the range 1 to 256, assigned to the application. Zero (0) is defined as always unsecured. See your security administrator about the security class conventions being used at your site. For more information on security classes, refer to the CA ADS Reference Section.

- **Response type** -- (Applies at definition time only) The default response type, **2 (local)**, specifies that you must explicitly make the response valid for any functions that use the response. You assign this response type to responses that are not used by many functions in the application. For example, it is easier to define ADD as a local response because you add it to only one function definition (DEPTMENU).
  A response that is available from many application functions is typically defined as a **1 (global)** response. A global response is added to each function definition as the function is defined. You later deselect the global response from the few functions that do not use the response. For example, you define BACK as a global response because it is used by all but one function in the application. When defining functions, it is easier to deselect BACK from one function than to explicitly select it for all other functions.
Note: It is often useful to list the global responses before local responses on the Response/Function List screen so that they are available for all function definitions.

- **Response execution** -- You can specify whether the invoked function is immediately executable or deferred. For more information on immediate and deferred execution, see CA ADS Reference Section.

- **Control command** -- You must specify the control command (for example, Transfer or Link) the response uses to invoke the specified function at runtime. For prototype applications, which don't perform any significant processing, you typically use the default control command, Transfer. When this command is used to pass control to another function, the runtime system releases resources being held for the original function. For more information on control commands, see CA ADS Reference Section.

**Responses for Department application**

The table below summarizes the specifications you will make for each of the five responses in the Department application. You can define these responses in any order, although it is often useful to define the global responses first. In the sample definition session shown in this section, these responses are defined in the order presented below.

<table>
<thead>
<tr>
<th>Response name</th>
<th>Description</th>
<th>Control key</th>
<th>Function invoked</th>
<th>Response type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACK</td>
<td>Return to menu</td>
<td>CLEAR</td>
<td>POP</td>
<td>Global</td>
</tr>
<tr>
<td>EXIT</td>
<td>Terminate application</td>
<td>PF9</td>
<td>QUIT</td>
<td>Global</td>
</tr>
<tr>
<td>ADD</td>
<td>Add a new department</td>
<td>PF4</td>
<td>ADDDEP</td>
<td>Local</td>
</tr>
<tr>
<td>MOD</td>
<td>Modify a department</td>
<td>PF5</td>
<td>MODDEP</td>
<td>Local</td>
</tr>
<tr>
<td>DEL</td>
<td>Delete a department</td>
<td>PF6</td>
<td>DELDEP</td>
<td>Local</td>
</tr>
</tbody>
</table>

To begin defining application responses, access the Response Definition screen by pressing [PF5] from the Response/Function List screen and define an application response.

The first two responses, BACK and EXIT, are used by most functions in the application. To make the functions easier to define later, you will define BACK and EXIT as **global responses**. When you define functions in ADSA, a global response is automatically available from each function. You can selectively remove a global function from functions when you define the functions.

You define the BACK response as shown:

**Defining the BACK response**

Response Definition

<table>
<thead>
<tr>
<th>Application name</th>
<th>Version</th>
<th>Response name</th>
<th>Function invoked</th>
<th>Description</th>
<th>Response type</th>
<th>Assigned key</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXAPPL</td>
<td>1</td>
<td>BACK</td>
<td>POP</td>
<td>Return to menu</td>
<td>1. Global</td>
<td>CLEAR</td>
</tr>
</tbody>
</table>

13-Jan-2018
After you press [Enter], ADSA adds the response definition if there are no errors, and then redisplays the Response Definition screen.

Defaults

ADSA fills in default values for the following Response Definition screen prompts if you have not entered another value:

- **Response type** returns the default value of 2 (Local). BACK is defined as a **global** response because it is available from many functions in the Department application. To specify that this response is global, overtype 2 with 1.

- **Response execution** returns the default value of 2 (Deferred)

- **Control command** returns the default value of 1 (Transfer).

When you press [Enter], the screen will redisplay with a confirming message if the definition is correct, or an error message if an error has been encountered.

⚠️ **Note:** If you press [Enter] after providing information, the screen will redisplay. If you press [PF5] after providing information **and there are no errors**, the appropriate Function Definition screen will be displayed. Since BACK and EXIT are responses that invoke system functions, the Function Definition screen will not be displayed.

Press [PF5] and define the EXIT response in the same way:

**Defining the EXIT response**

<table>
<thead>
<tr>
<th>Response Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application name: XXXAPPL</td>
</tr>
<tr>
<td>Response name: EXIT</td>
</tr>
<tr>
<td>Function invoked: QUIT</td>
</tr>
<tr>
<td>Description . . . . terminate application</td>
</tr>
<tr>
<td>Response type . . . . 1</td>
</tr>
<tr>
<td>Response execution . . . 2</td>
</tr>
<tr>
<td>Assigned key . . . . . PF09</td>
</tr>
<tr>
<td>Control command . . . . 1</td>
</tr>
<tr>
<td>3. Link 4. Return</td>
</tr>
<tr>
<td>5. Return continue 6. Return clear</td>
</tr>
<tr>
<td>7. Return continue clear 8. Transfer nofinish</td>
</tr>
</tbody>
</table>
Invoking nosave

Link nosave

These responses invoke system functions. You do not have to further define any system function.

**Defining the ADD response**

The ADD response invokes the function, ADDDEP. The response definition is as follows:

Response Definition

<table>
<thead>
<tr>
<th>Application name:</th>
<th>XXXAPPL</th>
<th>Version:</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response name:</td>
<td>ADD</td>
<td>Drop response (/)</td>
<td></td>
</tr>
<tr>
<td>Function invoked:</td>
<td>ADDDEP</td>
<td>Security class:</td>
<td>0</td>
</tr>
<tr>
<td>Description:</td>
<td>add department</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response type:</td>
<td>. . . . . 2 1. Global 2. Local</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response execution</td>
<td>. . . . 2 1. Immediate 2. Deferred</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assigned key:</td>
<td>. . . . PF04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADD is defined as a local response because it is available only from the DEPTMENU function in the Department application. You will make ADD valid for DEPTMENU when you define the DEPTMENU function in Step 6.

After you press [Enter], ADSA adds the response definition if there are no errors, and then redisplays the Response Definition screen.

When you are finished defining a response for the Department application, you define the function it invokes as described below. Pressing [PF5] will display the Function Definition screen.

**Step 6 Further Define the Application Functions**

Each function in an application represents a unit of work. For example, the DEPTMENU function lists available responses and allows the user to select a response. In this step, you will further define the following Department application functions:

- The **DEPTMENU menu function**, which displays a list of available responses (ADD, MOD, DEL, and EXIT)

- The **ADDDEP dialog function**, which allows the user to add a department record

- The **MODDEP dialog function**, which allows the user to modify an existing department record
The \textbf{DELDEP dialog function}, which allows the user to delete an existing department record.

\begin{itemize}
  \item \textbf{Note}: You do not have to define the POP or QUIT functions. POP and QUIT are system functions that have reserved names and meanings. ADSA automatically will add complete definitions for these functions when you associate them with the BACK and EXIT responses.
\end{itemize}

\section{Function Definition screens}

You define menu and dialog functions in the Department application by using one of several Function Definition screens:

\begin{itemize}
  \item \textbf{Function Definition (Dialog)} screen allows you to define dialog functions
  \item \textbf{Function Definition (Menu)} screen allows you to define menu functions
  \item \textbf{Function Definition (Program)} screen allows you to define program functions
\end{itemize}

For further information on program functions, see \textit{CA ADS Reference Section}.

Some of these screens are made up of multiple pages accessed through \texttt{[PF7]} and \texttt{[PF8]}. The appropriate screen will be displayed based on the initial function definition you gave on the \textit{Response/Function List} screen.

\section{Screen prompts}

Certain information will be presented to you when this screen is displayed:

\begin{itemize}
  \item \textbf{Application name} -- The name of the application will be displayed.
  \item \textbf{Version} -- The version number of the application will be displayed.
  \item \textbf{Function name} -- The name of the function as listed on the \textit{Response/Function List} screen will be displayed. ADDDEP, for each function.
  \item \textbf{Associated dialog} (dialog functions only) -- The name of the dialog associated with this function will be displayed. A dialog is an application component that typically displays a screen to the user and processes information.
\end{itemize}

\begin{itemize}
  \item \textbf{Note}: The dialogs that you name in this section are not yet defined; you will define dialogs for the Department application in Section10, "Defining Dialogs Using ADSC".
\end{itemize}

Use the \textbf{Function Definition} screen to define basic information on a function, such as:
A description -- You can specify a one-line description to help you identify the purpose of each function. This description is also available to the programmer (to place as a header on a map, for example) through the global record.

The responses that the user can select from the function. For example, as shown earlier in this section, ADD, MOD, DEL, and EXIT are valid from the DEPTMENU function. EXIT is a global response, so it is automatically available from DEPTMENU. You will use the Function Definition screen to make ADD, MOD, and DEL also available from DEPTMENU.

Specify how menu screens are to appear at runtime (resequencing of menu items and addition of header information). CA ADS uses your specifications to format and display menus at runtime.

For more information on these and other ADSA screens, see CA ADS Reference Section.

The Department application

The following table summarizes the specifications you will make for the functions in the Department application. You define one application function at a time, transferring between ADSA screens as necessary. Instructions for defining the DEPTMENU menu function and for defining the ADDDEP, MODDEP, and DELDEP dialog functions are presented below.

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Associated dialog</th>
<th>Function type</th>
<th>Valid responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDDEP</td>
<td>Add department</td>
<td>XXXDADD</td>
<td>1 (Dialog)</td>
<td>BACK EXIT</td>
</tr>
<tr>
<td>MODDEP</td>
<td>Mod department</td>
<td>XXXDUPD</td>
<td>1 (Dialog)</td>
<td>BACK EXIT</td>
</tr>
<tr>
<td>DELDEP</td>
<td>Del department</td>
<td>XXXDUPD</td>
<td>1 (Dialog)</td>
<td>BACK EXIT</td>
</tr>
<tr>
<td>DEPTMENU</td>
<td>Department menu</td>
<td>Not applicable</td>
<td>3 (Menu)</td>
<td>ADD MOD DEL EXIT</td>
</tr>
</tbody>
</table>

Dialog and menu functions are discussed separately below. Remember, however, that these screens will be displayed in the order in which you selected the functions on the Response/Function List screen. Pressing [PFS] will move you from a response definition to a function definition and then on to the next response definition.

Dialog functions

The functions, ADDDEP, MODDEP, and DELDEP, are all dialog functions. You named these dialog functions on the Response/Function List screen earlier in this section. At that time, ADSA automatically added skeleton definitions for the associated ADDDEP, MODDEP, and DELDEP functions. When you further define a dialog function, you enhance these skeleton function definitions.

Pressing [PFS] from the Response Definition screen for the response ADD brings you to the dialog function definition for dialog ADDDEP in the Department application.
You use the **Function Definition (Dialog)** screen to specify basic information for the dialog function, including a description, and valid responses for that function.

### Function description

When you display the Function Definition screen, the **Description** field will contain a value of **UNDEFINED** because you have not yet provided a description. A description must be provided before the application can be compiled.

### Valid responses

Valid responses are the responses (for example, BACK or EXIT) that a user can access directly from a function. You specify a valid response by entering a nonblank character opposite the response. When the screen is refreshed, a slash (/) is displayed.

A response that has been defined as **global** on the Response Definition screen will already be selected as a valid response for this function. You can deselect a response by spacing over the slash.

To make it easier for you to define dialog functions, dialogs that you name on the Function Definition screen do not yet have to be defined in the data dictionary. You will define dialogs for Department application dialog functions later in this section, in Section10, "Defining Dialogs Using ADSC".

### Defining the ADDDEP function

You specify basic information for the ADDDEP function as shown:

```
Function Definition (Dialog)

Application name: XXXAPPL  Version: 1
Function name: ADDDEP
Description . . . UNDEFINED

Associated dialog . . . XXXDADD  User exit dialog . . . . . .
Default response . . . . .

Valid

response(/)  Response Key  Function  Valid
response(/)  Response Key  Function

_  ADD  PF04  ADDDEP  _
  _  MOD  PF05  MODDEP
  _  DEL  PF06  DELDEP
  /  BACK  CLEAR  POP
  /  EXIT  PF03  QUIT

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

You specify basic information for the ADDDEP function as shown:
When you press [Enter], ADSA redisplays the Function Definition screen with an appropriate message. A confirming message is returned if there are no errors. In this case, you have successfully modified and enhanced the skeleton dialog function.

An error message is returned if ADSA detects any errors. In this case, use the message to determine the problem. You can type over any errors, and then press [Enter] again.

Press [PF5] to see the next Response Definition screen.

⚠️ **Note:** If you press [PF5] rather than [Enter] after providing information and there are no errors, the appropriate Response Definition screen (or Function Definition screen, if there are no more responses to be defined) is displayed immediately.

### Defining the MOD response

You can now define the MOD response.

Response Definition

Application name: XXXAPPL Version: 1
Response name: MOD Drop response (/)  _
Function invoked: MODDEP
Description . . . . . modify department Security class: 0
Response type . . . . . . 2  1. Global  2. Local
Response execution . . . 2  1. Immediate  2. Deferred
Assigned key . . . . . . . PF05
Control command . . . 1  1. Transfer  2. Invoke
  3. Link  4. Return
  5. Return continue  6. Return clear
  7. Return continue clear  8. Transfer nofinish
  9. Invoke nosave  10. Link nosave

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

### Defining the MODDEP function
Press [PF5] to see the Function Definition screen for the function that response MOD invokes, MODDEP.

You now can define the MODDEP dialog function by using the Function Definition screen:

Function Definition (Dialog)

Application name: XXXAPPL    Version: 1
Function name: MODDEP    Drop function (/) 
Description . . . modify department
Associated dialog . . . . XXXDUPD User exit dialog . . . . _______
Default response . . . . _______
Valid response(/)  Response Key  Function   Valid response(/)  Response Key  Function
_  ADD  PF04  ADDDEP  _    _______  _______
_  MOD  PF05  MODDEP  _    _______  _______
_  DEL  PF06  DELDEP  _    _______  _______
7  BACK  CLEAR POP  _    _______  _______
/  EXIT  PF09  QUIT  _    _______  _______
_    _______  _______  _    _______  _______

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

When you are finished defining MODDEP, press [PF5] to go directly to the next selected response.

Use the Response Definition and Function Definition screens to define the DEL response and the DELEMP function.

Defining the DEL response

Response Definition

Application name: XXXAPPL    Version: 1
Response name: DEL    Drop response (/) 
Function invoked: DELDEP
Description . . . delete department
Response type . . . . . 2  1. Global  2. Local
Response execution . . . 2  1. Immediate  2. Deferred
Assigned key . . . . . . PF06
Control command . . . . 1  1. Transfer  2. Invoke
   3. Link  4. Return
   5. Return continue  6. Return clear
   7. Return continue clear  8. Transfer nofinish
   9. Invoke nosave  10. Link nosave

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

Defining the DELDEP function

Function Definition (Dialog)
Once you have defined the responses and functions above, press PF5 to bring you to another Function Definition screen. There is one function as yet undefined: DEPTMENU. DEPTMENU is a menu function.

Menu functions

The last function listed in the table above is DEPTMENU, a menu function. You specified that it was a menu function on the Response/Function List screen in Step 3. To further define DEPTMENU, you must access the Function Definition screen by pressing [PF5] from the Response Definition screen.

Sample screen

You use the Function Definition (Menu) screen to specify basic information on a menu function. Each Function Definition (Menu) screen is made up of two pages. You can access the second page of the screen by pressing [PF8].

Menu screens
Menu screens are supplied by CA ADS; you do not have to write any statements to handle the display or operation of a menu. To tailor the appearance of a menu display, you can define a header to be displayed at the top of the menu screen. This header can include a title, instructions, or any other appropriate text.

You specify basic information for function DEPTMENU as shown:

**Defining the DEPTMENU function**

Function Definition (Menu)    Page 1 of 2

- Application name: XXXAPPL    Version: 1
- Function name: DEPTMENU
- Description: department menu
- Associated dialog
- Default response
- User exit dialog
- Use signon menu (/)
- Menu defined by: 2 1. User 2. System
- Description length: 1 Long (28) 2. Short (12)
- Responses per page: 15
- Number of heading lines (0-3): 2
- Heading line text

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

After you press [Enter], ADSA redisplays the Function Definition screen with a confirming message. An error message is returned if ADSA detects any errors. In this case, use the message to determine the problem. You can type over any errors, and then press [Enter] again.

**Second page of Function Definition (Menu)**

You use the second page of the Function Definition (Menu) screen to specify the responses (such as ADD) that a user can access directly from the function. You also specify the sequence that the response will be displayed on the menu. You access the second page by pressing [PF8]. Second and subsequent pages are response sequence screens.

Access the second page and make the ADD, MOD, DEL, and EXIT responses valid from the DEPTMENU function as shown:

Function Definition (Menu)    Page 2 of 2

- Application name: XXXAPPL    Version: 1
- Function name: DEPTMENU
- Valid resp. Seq. # Response Key Function Valid Resp. Seq. # Response Key Function
  _        __ BACK CLEAR POP _        __ __ __ __ __ __ __
  / 400 ___ EXIT PF09 QUIT _ __ __ __ __ __ __ __
  / 100 ___ ADD PF04 ADDDEP _ __ __ __ __ __ __ __
  / 200 ___ MOD PF05 MODDEP _ __ __ __ __ __ __ __
Specifying menu sequence

You can use this screen to specify the sequence in which valid responses are displayed on the menu screen.

You do this by entering sequence numbers for each valid response.

If you want to change the sequence numbers, you can change those numbers:

100 ADD
200 DEL
300 EXIT
150 MOD

To display MOD between ADD and DEL, you would change 400 to some value between 100 (ADD) and 200 (DEL).

For example: 101 or 150.

Response sequence numbers displayed on the Function Definition screen are not displayed to users.

Inhibiting response display

You can also use this screen to inhibit the display of valid responses on the menu screen.

You do this by replacing the sequence number for the response with 0 (zero):

100 EMPINFO
200 DEPTINFO
000 SALARIES
400 EXIT

To inhibit display of the SALARIES response, you would replace 300 with 0 (zero).

Invisible responses can still be accessed by any user who knows the response name or control key. To actively restrict responses, see information on security in the CA ADS Reference Section.

For more information on other uses of the Function Definition (Menu) screen, see CA ADS Reference Section.

The DEPTMENU function is now fully defined.

After you finish defining functions and responses for the Department application, press [PF5] to return to the Response/Function List screen. Processed selections on the Response/Function List screen will be de-selected; unprocessed selections will still be selected and are accessed when you press [PF5].

It is helpful to re-access the Response/Function List screen to remind yourself where you are in the definition process.

Press [PF5] again to go to the Global Records screen.
Press [PF5] again to go on to the Task Codes screen where you define a task code for the application. (Alternatively, you can press [PF3] from the Response/Function List screen to return to the Main Menu and choose option 4.)

**Step 7 Define a Task Code**

A task code defines an entry point into an application. At run time, developers and users can execute an application by using the application’s task code. An application can have more than one task code; each one can be associated with a different function in the application.

As an application developer, you must define at least one task code for an application before you can successfully create a load module for the application. When you define a task code, you name:

- **A task code** (for example, XXXDEPT) that a user can supply to invoke the application
- **The function** (for example, DEPTMENU) to be executed first when a user supplies the associated task code

You use the **Task Codes** screen to define task codes for an application. You can access the Task Codes screen from the Response/Function List screen by pressing [PF5].

Define a task code as shown:

**Defining the task code**

<table>
<thead>
<tr>
<th>Application name: XXXAPPL</th>
<th>Task Codes</th>
<th>Page 1 of 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version: 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task Code</th>
<th>Function</th>
<th>Drop (/)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. xxxdept</td>
<td>deptmenu</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

After you press [Enter], ADSA redisplays the Task Codes screen with a confirming message. An error message is returned if ADSA detects any errors. In this case, use the message to determine the problem. You can type over errors, and then press [Enter] again.

When you have defined a task code for the Department application, you can create a load module for the application as described below.
Step 8 Compile the Application

When you compile an application, ADSA creates a load module that incorporates all of your specifications. You compile an application by selecting the **Compile** activity from the action bar on the Main Menu screen.

You compile an application from the Main Menu screen. To get to the Main Menu screen from the Task Codes screen, press [PF5].

**Compiling the application**

To compile the application, position the cursor on the **Compile** item on the action bar and press ENTER. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing `compile` on the command line and pressing [Enter]

Once you have displayed the **Compile** action item, choose 1 and press [Enter] to compile the application.

After you press [Enter] to compile the application, ADSA displays messages to indicate whether the application has been compiled successfully.

If you receive an error message you can display diagnostic information by selecting the **View messages** option from the **Compile** activity on the action bar. Based on this information, you can correct the application and then compile it again.
Exit from ADSA

You can return directly to DC/UCF from the Main Menu by pressing PF3. Alternatively, you can use the **Switch** activity on the action bar of the Main Menu screen to transfer to another development tool.

In this sample session, you'll exit to DC/UCF so you can execute your application structure. Press [PF3] to exit.

**Note:** If you leave ADSA without successfully compiling the current application definition, ADSA saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the application definition. To enable them to access the definition, specify the **Release** option from the **Modify** activity on the action bar on the Main Menu.

After you exit from ADSA, you can execute your application as described below.

Optionally Execute the Application

In the previous steps, you defined an executable application structure by using ADSA screens. At this stage, your menu and system functions are fully defined. Your application prototype is not fully operational, however, until you associate the dialog functions in your application with executable dialogs.

In a typical development environment, application developers don't execute an application until the prototype is complete. If you would like to execute your application to see what you've already created, you can invoke and test the application as described below.

Invoke the application

CA ADS applications execute under the CA ADS runtime system. To invoke the Department application from DC/UCF, you can enter the task code for the runtime system, followed by the application task code.

For example, assume that ADS is the task code for the runtime system and XXXDEPT is the task code for the Department application. You can invoke the Department application as shown:

```
ads xxxdept
```

Sample task code for the Department application

Sample task code for the runtime system

[Enter]

The DEPTMENU screen is the first screen displayed.
Alternatively, you can enable developers and users to simultaneously invoke the runtime system with an application. To do this, you associate the application task code with the runtime system in either of the following ways:

- **At system runtime** you issue a DCMT VARY DYNAMIC TASK command while using DC/UCF to dynamically associate a task code with the runtime system. Use of the DCMT VARY DYNAMIC TASK command is shown in , later in this section.

- **At system-generation time** you use the TASK statement in the system definition to associate the application task code with the CA ADS runtime system.
  This procedure typically is used for a production application. For more information, see *CA IDMS System Generation Section*

After you invoke the Department application from DC/UCF, you can test out features that you've already implemented, as described below.

**Test current features**

Using ADSA, you made the ADD, MOD, DEL, and EXIT responses valid from the DEPTMENU function in Step 6. Of these responses, EXIT is the only response that is associated with a fully defined function.

To see how the EXIT response works, select EXIT from the menu screen in any of the following ways:

```
DIALOG:          DATE: 08/19/99
                PAGE: 1 OF: 1
                NEXT PAGE: 
DEPARTMENT INFORMATION APPLICATION
MAIN MENU

_ ADD (PF4) ADD A NEW DEPARTMENT
_ MOD (PF5) MODIFY A DEPARTMENT
_ DEL (PF6) DELETE A DEPARTMENT
x EXIT (PF9) TERMINATE APPLICATION
```

**RESPONSE:**

To test out other features of the Department application, you can invoke the application again, as described earlier in *Invoke the application (see page 210).*

You can try pressing a control key (such as &pf4). that is not associated with a response on this menu. The runtime system automatically detects undefined control keys and returns the following message:

```*** UNACCEPTABLE RESPONSE. PLEASE TRY AGAIN ***```

**Requesting a function that requires further definition**
You can try the ADD, MOD, and DEL responses if you want. Since the associated ADDDEP, MODDEP, and DELDEP functions still require further definition, selecting ADD, MOD or DEL will cause the application to terminate. In this case, the runtime system displays the Dialog Abort Information screen:

**Dialog Abort Information screen**

```
CA ADS RELEASE nn.n             *** DIALOG ABORT INFORMATION ***     ABRT
DC171028 APPLICATION NOT EXECUTED. DIALOG LOAD MODULE XXXDADD MISSING

DATE....: 91.078      TIME....: 10:30:51.30      TERMINAL.....: LV35003
ERROR OCCURRED IN DIALOG.......: XXXDADD
   AT OFFSET.......:
   IN PROCESS......: VERSION: 0
   AT IDD SEQ NO. : 00000000

SEQUENCE
NUMBER: 00000000
00000000
00000000
00000000

HIT ENTER TO RETURN TO DC OR ENTER NEXT TASK CODE:
```

The **Dialog Abort Information** screen is particularly useful when you are developing and debugging process logic for dialogs. At that time, this screen can help you determine where in a module of process code the dialog fails.

The display of this diagnostic screen can be disabled when the application is ready for final release.

### Summary

#### Creating the executable structure

In this section, you used ADSA screens to create the executable structure for the sample Department application. The structure includes:

- **Responses**, which define the possible runtime paths available to users of the application
- **Functions**, which define the activities that users can perform while using the application
- A **task code** that defines an entry point into the application and allows users to invoke the application

You built the application structure by establishing relationships between responses and functions as you defined them:

- **For each response**, you named the function to be invoked by the response.
- **For each function**, you named the responses to be valid from the function. At runtime, the user can select any of the valid responses from the function.
For menu function DEPTMENU, you also specified how options for the user and the menu's title are to appear on the menu screen at runtime. For each of the application's dialog functions, you also named the executable component (that is, the dialog) to be executed at runtime when the user invokes the associated function. You will actually define these dialogs in Section 10, "Defining Dialogs Using ADSC".

Creating the load module

When you finished defining the structure of the Department application, you created a load module for the application. You created this executable load module without explicitly writing any lines of procedural code. By using ADSA screens, you have implicitly coded all potential flow of control for the application.

Even at this early stage in the application development cycle, your application contains fully executable components, such as function DEPTMENU. Your dialog functions (ADDDEP, MODDEP, and DELDEP) are not yet developed fully, so you cannot execute them. As soon as you develop the ADDDEP, MODDEP, and DELDEP dialog functions, your application will be fully executable.

The first step in developing dialogs is to create screens that the dialogs will display to users. You will create the screen display for the ADDDEP, MODDEP, and DELDEP dialog functions in the next section.

Defining a Screen Display Using MAPC

In the previous section, you used ADSA to define the structure of the sample Department application. As another step in creating an application, you define screen displays by using the map compiler (MAPC). This section provides instructions for defining the XXXMAP screen display for use in the sample Department application.

Note: When creating your map, you can substitute your initials for the XXX in the map name.

This section includes:

- An overview of how maps are used in the CA ADS environment
- Instructions for defining maps for the sample Department application
- A summary of what you've accomplished in this section

What is a Map: A predefined screen display used by dialogs in an application. At runtime, dialogs use maps to interact with users. For example, the sample map XXXMAP that you create in this section is used to display existing department records to users for modification or deletion. XXXMAP also allows users to add new department records. The XXXMAP layout is shown in the diagram below. This sample map allows users to input and display a department's ID number and name, and the ID number of the department head.

XXXMAP Layout
CA IDMS - 19.0

FUNCTION: __________

DEPARTMENT INFORMATION

DEPARTMENT ID ........: ___
NAME ........: ________________________________________________
HEAD ID ..: ___

NEXT RESPONSE: __________

-------------------------------------------------------------------------------------------------

Defining a Map

You define the layout of a map by defining individual fields on the map. You can define two types of fields:

- **Literal fields** display unchanging literal strings.
  Titles, instructions, and prompts often are defined as literal fields. For example, the DEPARTMENT INFORMATION title shown in the map layout is a literal field.

- **Variable fields** display stored values and allow users to store new values at runtime.
  For example, the field to the right of the DEPARTMENT ID field on the map layout is a variable field. At runtime, this field displays the ID number for a department record. Additionally, the user can type a new id number into this variable field to modify or add a department record. The last field on the screen, the message field, is also a variable field. At runtime, this field displays a message (message field is blank in this sample screen).

You will use the **online mapping facility (MAPC)** to define map XXXMAP shown in the map layout.

A typical MAPC screen is shown below:

```
CA IDMS/DC Online Map Compiler

Map name ...........: __________
Map version ........: ___
Dictionary name .....: __________
Dictionary node .....: __________
Screen .............: 1. General options
                    2. Map-Level help text definition
                    3. Associated records
                    4. Layout
                    5. Field definition

Copyright (C) 1999 CA, Inc.

Command ===> Enter F1=Help F3=Exit F10=Action
```
Define Maps Instructions

You use MAPC to define maps. To define XXXMAP in this section, perform the following steps:

- Step 1 Invoke MAPC (see page 215)
- Step 2 Name the map (see page 215)
- Step 3 Name the records (see page 217)
- Step 4 Create the map with the autopaint facility (see page 220)
- Step 5 Modify the map layout (see page 222)
- Step 6 Select fields for further definition (see page 228)
- Step 7 Edit variable fields (see page 229)
- Step 8 Edit literal fields (see page 232)
- Step 9 Compile the map (see page 237)
- Exit from MAPC (see page 238)
- Optionally display the map (see page 238)

After you compile the map, you can exit from MAPC and optionally display the map you have just defined. Steps for defining map XXXMAP are presented below.

Step 1 Invoke MAPC

You can invoke MAPC from CA IDMS/DC or DC/UCF by specifying the task code for MAPC (for example, MAPCT) in response to the prompt presented by DC/UCF. For example, you can invoke MAPC from CA IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
mapct
```

Press the ENTER key to input the - [Enter] task code for MAPC.

For more information on task codes for CA ADS development tools, see Application Development Tools (see page 180). MAPC begins by displaying the Main Menu screen, on which you specify basic information on a map. Use the Main Menu screen to begin defining map XXXMAP, as described in Step 2.

Step 2 Name the map

The first screen in an MAPC session is the Main Menu screen. A sample Main Menu is shown below:

**Sample Main Menu Screen**

```
Add  Modify  Compile  Delete  Display  Switch
-----------------------------------------------------------------------
CA IDMS/DC Online Map Compiler

Map name ... ... : ........
Map version ... : ... ___
```
Screen Prompts

When you begin a map definition, you typically enter information after one or more of the following Main Menu screen prompts:

- **Map name** -- You must supply a map name. The name you specify must be unique among all programs. (For example, it cannot be the same name as a dialog.)

- **Map version** -- You type the version number of the map after the Map version prompt. If not specified, the map version defaults to the version number specified as a dictionary option.

- **Dictionary name** -- You must specify the same dictionary (if any) as you specified for your application definition in Step 2: Name the Application (see page 190). The correct dictionary name may already be displayed in this field.

- **Dictionary node** -- You must specify the same dictionary node (if any) as you specified for your application definition in Section 7. The correct dictionary node may already be displayed in this field.

You specify basic information on map XXXMAP on the Main Menu screen.

**Defining the XXXMAP Map**

You can use the tab key to move the cursor quickly and easily between prompts. Begin defining your application on the Application Definition screen as shown below:

```
Add Modify Compile Delete Display Switch
```

```
CA IDMS/DC Online Map Compiler
CA, Inc.
```

```
Map name . . . . . . . . . . . xxxmap
Map version . . . . . . . . . 1
Dictionary name . . . . . . demo
Dictionary node . . . . . . . . . . . . 

Screen . . . . . . . . . . .
```

```
1. General options
2. Map-Level help text definition
3. Associated records
4. Layout
5. Field definition
```

Copyright (C) 1999 CA, Inc.
Adding the Map

To add the map, position the cursor on the Add item on the action bar and press [Enter]. You can position the cursor on Add by:

- Tabbing to Add and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to Add and pressing [Enter]
- Typing add on the command line and pressing [Enter]

Once you have displayed the Map action item, press [Enter] to add the map to the dictionary. After you press [Enter], the action is confirmed.

MAPC redisplays the Main Menu screen with a message:

- Map XXXMAP version 1 has been added is returned when your definition contains no errors.
- An error message is returned if MAPC detects any errors. Read the message to see what problem has occurred. You can type over any errors and then press [Enter] again.

After you provide basic information on the map, you can specify the records you want to use on the map.

Step 3 Name the records

The Associated Records screen is used to enter the schema or work records to be used by the map, and optionally specifies role names for records.

The auto paint feature is invoked from this screen. The auto paint feature automatically paints the map based on the elements you select.
You access the Associated Records screen from the Main Menu by entering the number 3 next to the Screen prompt and pressing [Enter].

Add Modify Compile Delete Display Switch

CA IDMS/DC Online Map Compiler

CA, Inc.

Sample Associated Records Screen

Map name: XXXMAP  Version: 1

Record name  Version  Role name  Drop (/)
1 __________________________  __________________________  _
2 __________________________  __________________________  _
3 __________________________  __________________________  _
4 __________________________  __________________________  _
5 __________________________  __________________________  _
6 __________________________  __________________________  _
7 __________________________  __________________________  _

Screen Prompts

On the Associated Records screen, you name the records that will be used on the map:

- **Record name** -- You associate existing database and work records with the map by naming the records in the lines below the Record name prompt.
  Associating a record with a map allows the map to display and store data for that record. Records associated with maps sometimes are referred to as map records to indicate that the records define data used by the map.
  Records contain record elements, which define data. In relational terminology, a record is a data table that contains columns defining data to be stored. An example of a record is the
DEPARTMENT database record. One of the elements in the DEPARTMENT record is DEPT-NAME-0410. This element stores department names. The layout of the sample DEPARTMENT record is provided in.

- **Version** -- You type the version number of the record below the Version prompt. For example, the demonstration database at your site might have a few different versions of the DEPARTMENT record, each reserved for specific testing or development purposes. In this case, each different version of the record has a unique version number (for example: 1, 2, or 100). To ensure that MAPC uses the correct version of the record, you specify the record version number along with the record name.

### Records Associated with XXXMAP

The following table lists the records you will associate with map XXXMAP.

<table>
<thead>
<tr>
<th>Record</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT</td>
<td>A database record in the demonstration database. DEPARTMENT includes elements for department ID number, department name, and the employee ID of the department head.</td>
</tr>
<tr>
<td>Version: 1 *</td>
<td></td>
</tr>
<tr>
<td>ADSO-APPLICATION-GLOBAL-RECORD</td>
<td>A special CA ADS record that contains information on the application at runtime. For example, this record includes a record element that at runtime contains the name of the currently executing function.</td>
</tr>
<tr>
<td>Version: 1</td>
<td></td>
</tr>
</tbody>
</table>

* A different version of the DEPARTMENT record may be provided for use at your site.

### Associating Records with XXXMAP

<table>
<thead>
<tr>
<th>Map name: XXXMAP Version: 1</th>
<th>Associated Records</th>
<th>Page 1 of 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 department______________________ 1 _____</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.ct ___________________________ _</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 adso-application-global-record__ 1 ____</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.ct ___________________________ _</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>
</tbody>
</table>

DC366601 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd  F9=Autopaint
After you have entered the names of the records on the Associated Records screen, press [PF9] to use the auto paint facility to create the map.

### Step 4 Create the map with the autopaint facility

You can create a map either sectionly or through the autopaint facility.

To create a map sectionly, you would name the records on the Associated Records screen and then place each literal and variable field explicitly on the map using the Layout screen. Then you would further define each variable field to associate a record element with that field.

For further information, see *CA IDMS Mapping Facility Section*.

The autopaint facility quickly creates a standard map layout based on the records you have named on the Associated Records screen. The autopaint facility is useful for maps which require little or no explicit screen placement.

You will use the autopaint facility to create the XXXMAP for the Department application.

The first step in using the autopaint facility to create a map is to go to the Automatic Screen Painter screen to identify the fields you want to have displayed on the map. Here you determine what record elements will be displayed on the screen.

A **record element** is a data definition that is contained in a record.

For example, the DEPARTMENT record contains a record element for a department's name, DEPT-NAME-0410.

If the dialog retrieves a department record from the database, the record is temporarily stored in variable storage. The record name in variable storage is automatically displayed when the map is displayed. If the user enters a valid department name in this map field, the data is automatically moved into variable storage. It can then be saved in the database or used to access other data, depending on the dialog code.

To access the Automatic Screen Painter screen, press [PF9] from the Associated Records screen.

**Sample Automatic Screen Painter Screen**

<table>
<thead>
<tr>
<th>Map name: XXXMAP</th>
<th>Automatic Screen Painter</th>
<th>Page 1 of 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select (/)</td>
<td>Version: 1</td>
<td>Element Level and Name Occurs</td>
</tr>
<tr>
<td></td>
<td>01 DEPARTMENT VERSION 0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02 DEPT-ID-0410</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02 DEPT-NAME-0410</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02 DEPT-HEAD-ID-0410</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 ADSO-APPLICATION-GLOBAL-RECORD VERSION 0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-APPLICATION-NAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-CURRENT-FUNCTION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-NEXT-FUNCTION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-CURRENT-RESPONSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-DEFAULT-RESPONSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-TASK-CODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-EXIT-DIALOG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-PRINT-DESTINATION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-DATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 AGR-USER-ID</td>
<td></td>
</tr>
</tbody>
</table>
DC365503 Select the fields that are to appear on the screen

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

Screen Prompts

On the Automatic Screen Painter screen, you can select the fields that will be used on the map. The fields will be listed under the appropriate record name. Use a nonblank character to select the fields you want displayed.

- **Element level and name** -- Each element within a record listed on the Associated Records screen will be displayed. Each element is associated with its level number as defined in the dictionary.

- **Occur** -- If the element is a repeating element, you can define the occurrence number in this column.

There can be multiple pages of elements. Press [PF8] to continue to the next page.

Fields on XXXMAP

The following table lists the records and fields to be displayed on XXXMAP.

<table>
<thead>
<tr>
<th>Record</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT</td>
<td>DEPT-ID-0410</td>
</tr>
<tr>
<td></td>
<td>DEPT-NAME-0410</td>
</tr>
<tr>
<td></td>
<td>DEPT-HEAD-ID-0410</td>
</tr>
<tr>
<td>ADSO-APPLICATION-GLOBAL-RECORD</td>
<td>AGR-CURRENT-FUNCTION</td>
</tr>
</tbody>
</table>

Note:

The database does not have to be defined before you create the prototype map layout. If the database were *not* already defined, you would define all literal fields sectionly. (See )

Selecting Fields for Use with XXXMAP

Map name:  XXXMAP  Version:  1  Page  1  of  3

<table>
<thead>
<tr>
<th>Automatic Screen Painter</th>
<th>Page 1 of 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map name:  XXXMAP  Version:  1  Page  1  of  3</td>
<td></td>
</tr>
<tr>
<td>Select (/)</td>
<td>Element Level and Name Occurs</td>
</tr>
<tr>
<td>01 DEPARTMENT VERSION 0001</td>
<td></td>
</tr>
<tr>
<td>/ 02 DEPT-ID-0410</td>
<td></td>
</tr>
<tr>
<td>/ 02 DEPT-NAME-0410</td>
<td></td>
</tr>
<tr>
<td>/ 02 DEPT-HEAD-ID-0410</td>
<td></td>
</tr>
<tr>
<td>/ 01 ADSO-APPLICATION-GLOBAL-RECORD VERSION 0001</td>
<td></td>
</tr>
<tr>
<td>03 AGR-APPLICATION-NAME</td>
<td></td>
</tr>
<tr>
<td>/ 03 AGR-CURRENT-FUNCTION</td>
<td></td>
</tr>
<tr>
<td>03 AGR-NEXT-FUNCTION</td>
<td></td>
</tr>
<tr>
<td>03 AGR-CURRENT-RESPONSE</td>
<td></td>
</tr>
<tr>
<td>03 AGR-DEFAULT-RESPONSE</td>
<td></td>
</tr>
<tr>
<td>03 AGR-TASK-CODE</td>
<td></td>
</tr>
<tr>
<td>03 AGR-EXIT-DIALOG</td>
<td></td>
</tr>
<tr>
<td>03 AGR-PRINT-DESTINATION</td>
<td></td>
</tr>
<tr>
<td>03 AGR-DATE</td>
<td></td>
</tr>
<tr>
<td>03 AGR-USER-ID</td>
<td></td>
</tr>
</tbody>
</table>
DC365503 Select the fields that are to appear on the screen

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

After you have selected the names of the fields on the Automatic Screen Painter screen, press [PF5] to paint the map.

⚠️ **Note:** You can press [Enter] first if you want the screen to be redisplayed to check your choices.

If there are multiple pages of elements, you would use [PF8] to move to subsequent screens for further selection.

The autopainted screen will be displayed on the Layout screen. Now you can modify the placement of the fields and request that some fields be further defined.

**Step 5 Modify the map layout**

You use the Layout screen to modify the layout of fields on a map. There are two types of fields on the screen:

- **Map literal fields** display predefined literal strings at runtime.
- **Map variable fields** display stored values and allow users to store new values at runtime.

When you modify map XXXMAP in this section, you will:

- Add new literal fields
- Modify literal fields
- Change the placement of variable and literal fields

**Accessing the Layout Screen**

Once you have pressed [PF5] from the Automatic Screen Painter screen, the Layout screen is displayed to you with the automatically-created map presented. Fields at the bottom of the screen show key functions and a scale.

To reveal the hidden portion of the screen, press [PF8].

**Sample Layout Screen with Additional Fields Displayed**

```plaintext
;DEPT-ID-0410 ;__*_
;DEPT-NAME-0410 ;_________________________________________*
;DEPT-HEAD-ID-0410 ;_________________________________________*
;AGR-CURRENT-FUNCTION ;_________________________________________*
```
Note: Although you did not specify a response or message field, ADSA provided both. Since you associated the global record with this map, ADSA provided AGR-MAP-RESPONSE and AGR-MESSAGE as the response and message variable fields. If you had not associated the ADSO-APPLICATION-GLOBAL-RECORD with this map, ADSA would have provided $RESPONSE and $MESSAGE as the response and message variable fields.

Notice that the field mark for the message field is in column 80 so that the message will begin in column 1 of the following line.

Press [PF11] to reveal the alternate PF key set used for tailoring the screen.

Sample Layout Screen with Alternate Set of PF Keys Displayed

;DEPT-ID-0410 ;____ *
;DEPT-NAME-0410 ;___________________________ *
;DEPT-HEAD-ID-0410 ;____ *
;AGR-CURRENT-FUNCTION ;________ *

When you first display the Layout screen for an existing map, each field on the map is preceded by a start-field character, as shown above. While using the Layout screen, you can select a field for editing by:

- Pressing [PF2] while on the field you want to select (with the main set of function keys displayed at the bottom of the screen)
  or

- Typing a select-field character (%) in place of the start-field character for the field.

You use the start-field and select-field characters based on the following sectionlines:

- The start-field character (default is ; or {) defines the start of a field on the Layout screen.
The **select-field character** (default is %) defines the start of a field and simultaneously selects the field for editing.

You use the Layout screen to modify the layout of map XXXMAP and to select fields on the map for further editing:

**Selecting Multiple Fields to Edit**

You can mark two fields with [PF2] to select either all literal fields or all data fields in the area bounded by the two fields you mark.

If the first field you marked with [PF2] was a literal field, all literal fields between the two marked fields are selected. If the first field you marked with [PF2] was a data field, all data fields between the two marked fields are selected.

**Adding Literal Fields**

You place each field on the Layout screen in the following manner:

1. **Position the cursor** by using any of the cursor movement keys.
2. **Type a start-field character** for each literal and variable field.

**Note:** In this document, the start-field character is shown as a ";". Do not confuse this with a semi-colon.

On the Layout screen, the start-field character signals the start of a field. For example:

Sample start-field character.

;sample literal field

The field itself starts in the column that immediately follows the start-field character.

At runtime, the start-field character is not shown to users with 3270-type terminals. Instead, each field is preceded by a nondisplayable **attribute byte**. The attribute byte specifies the runtime characteristics of the field, such as input restrictions and display intensity.

The default start-field character for the Layout screen is:

- For **IBM-type terminals**, the field mark character (;)

**Note:** The field mark is not the same as the semicolon character. To type a field mark, you press the FIELD MARK key.
• For **Siemens-type terminals**, the left brace character ({)

\[\text{Note:} \text{ The start-field character for the Layout screen is defined at system-generation time, and can vary from site to site.}\]

• **Type the literal string** (for literal fields only) after the start-field character.

\[\text{Note:} \text{ To add a variable field, you need only type the start-field character. (See .)}\]

Changing the Content of a Literal Field

To change the content of an existing literal field, type characters or spaces over the fields that you want to change.

\[\text{Note:} \text{ Use the ERASE EOF key only if you want to erase everything that can be seen on the Layout screen starting at the current cursor position.}\]

Moving Fields, Lines, and Blocks

You can move fields or groups of fields:

• **Moving one field** -- Move the cursor to the field you want to move and press [PF2] with the alternate set of function keys displayed at the bottom of the screen. This marks the field.

• **Moving a line** -- Move the cursor to the starting position of the block you want to move and press [PF2] twice. This marks the line.

• **Moving a block** -- Move the cursor to the starting position of the block you want to move and press [PF2]. Move the cursor to the ending point of the block and press [PF2] again. This marks a block.

Move the cursor to the desired target location for the field, line, or block, and press [PF4].

Deleting Fields, Lines, and Blocks

Mark the field, line, or block. Then press [PF5].

Copying Fields, Lines, and Blocks

Mark the field, line, or block. Then press [PF3].

When you copy literal or variable fields, the complete definition of the literal or variable field (including attributes, etc.) is copied. Copying a data field that occurs increments the subscript to the next available value.
Modifying the Map Layout for XXXMAP

For map XXXMAP, you selected four variable fields that are now shown on the Layout screen. These are associated with four literal fields. Each literal and variable field is shown with the **field mark (;)** used as the start-field character.

;DEPT-ID-0410 ;____*
;DEPT-NAME-0410 ;___________________________________________*
;DEPT-HEAD-ID-0410 ;____ *
;AGR-CURRENT-FUNCTION ;________*

You are going to modify this map so that it looks like the following layout:

FUNCTION: ________

DEPARTMENT INFORMATION

DEPARTMENT ID......: _____*
NAME......: ____________________________*
HEAD ID..: _____*

NEXT RESPONSE: _

Your modifications involve:

- Changing the AGR-CURRENT-FUNCTION literal field to FUNCTION
- Moving the FUNCTION literal and variable fields
- Adding a title
- Modifying the remaining literal fields

Modifying the FUNCTION Fields for XXXMAP

The AGR-CURRENT-FUNCTION fields (literal and variable) should be placed in the upper left corner of the screen and the literal changed to FUNCTION according to the sample screen.

To change the FUNCTION field:
1. Overtype the literal field (leaving the field mark) with the word FUNCTION

2. Mark the variable field

3. Move the variable field closer to the FUNCTION literal field

4. Mark the line containing the literal and variable fields

5. Move the line to the upper left corner

Add the Title

1. Place the cursor where you want the title to begin.

2. Type a field mark

3. Type the title, DEPARTMENT INFORMATION

Modify the Remaining Literal Fields

There are three literal fields relating to the DEPARTMENT record. Modify these fields so that they match the XXXMAP screen shown above.

1. Overtype the literal fields with the appropriate words

2. Mark each field that needs to be moved

3. Move the field

Remember that there is more room for screen layout hidden at the bottom of the screen. To see this hidden area, press [PF8].

The completed screen should look like the one below.

;FUNCTION: ;

;DEPARTMENT INFORMATION

;DEPARTMENT_ID ........:;
;NAME ........:;
;HEAD_ID ..:;

;NEXT_RESPONSE: ;

After you press [Enter], MAPC redisplays the Layout screen so that you can inspect the screen for errors. At this point, it is a good idea to verify that:

- You have preceded each literal field with a start-field character.
You have defined each variable field with a start-field character, including the field that starts on the bottom right-hand margin of the screen and wraps around to the last line on the screen.

Correcting Errors

If you find any mistakes in the map layout, you can correct the Layout screen in either of the following ways:

- To change a few fields, type over the characters that you want to change and press [Enter] again.
- To erase all fields that you just placed on the screen, press the CLEAR key. If you press CLEAR, you must then place fields on the Layout screen again, as described earlier.

When you are satisfied with the Layout screen, press [PF11] to return to the main keys. You can now go on to Step 6, where you will select fields for further definition.

Step 6 Select fields for further definition

In Step 5, you modified the position of fields on your map and redefined some of the literal fields.

At this point, literal fields (for example, DEPARTMENT ID) are fully defined, although you can modify their definitions at any time.

Variable fields may not be fully defined. In this step, you will select fields for further definition. You will edit the field definitions in Steps 7 and 8.

Selecting Fields

While on the Layout screen, you can select fields for further definition. To do this, you press [PF2] once while the cursor is on the field you want to select. Pressing [PF2] marks a field for selection. (Alternatively, you can overtype the start-field character with a percent sign - %.)

Select XXXMAP Fields

The XXXMAP fields you need to further define are:

- The FUNCTION variable field
- The DEPARTMENT INFORMATION literal field
- The DEPARTMENT ID variable field
- The NAME variable field
- The HEAD ID variable field
- The message variable field

;FUNCTION: %________
%DEPARTMENT INFORMATION

;DEPARTMENT ID .......: %__ *
;NAME ........: %------------------------------------------*
;HEAD ID ..: %__ *
Select the fields for further definition. Then press [PF5] to continue to the Literal Definition and Field Definition screens shown below. MAPC will bring up the appropriate Literal Definition or Field Definition screen depending on the fields you selected on the Layout screen. Pressing [PF5] will bring you to the next definition screen in order of your selection.

This section separates the discussion of variable and literal fields, but remember that MAPC will intermix the two.

Step 7 Edit variable fields

In this step, you will edit a variable field's definition to determine what characteristics the field will have at runtime. You access the Field Definition screen from the Layout screen by pressing [PF5] after you have selected fields from the screen, or by pressing [PF5] from another definition screen (either Field or Literal Definition).

There are seven pages of data field screens. Navigate through these pages using [PF7] or [PF8], or move directly to the desired page by overtyping the page number.

Sample Field Definition Screen - page 1

Field Definition
Map name: XXXMAP Version: 1
;FUNCTION: ;________*

Field at row 1 column 1 Drop field (/) _

Element name: AGR-CURRENT-FUNCTION Subscript
In record ADSO-APPLICATION-GLOBAL-RECORD Version 1

Edit Picture X(8)

Display intensity 2 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action

Unprotected (/) . . . . . . . / Required (/) . . . . . . . .
Automatically edited (/) / Skipped by tab key (/) _

DC366004 Specify the variable field and any attributes
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

Notice that the Field Definition screen shows the field to be defined plus two scales to help you position the field.
You use the Field Definition screen to edit one map field at a time. After you edit a field (using [PF8] if you need to go to further pages), press [PF5]. MAPC displays either the Field Definition screen or the Literal Definition screen for the next field that you've selected. When you've edited all selected fields, MAPC returns you to the Main Menu screen.

Note: Pressing [Enter] on any screen causes it to redisplay so that you can make changes if you want.

Prompts

In this step, you use the following prompts on page 1 of the Field Definition screen:

- **Element name** -- Name of a record element or special system field to be displayed and input in the variable map field.
  When you autopainted the map, you selected the record elements you wanted displayed in the variable fields on that map (such as the department-name record element for the variable map field that follows the literal FUNCTION field). These element names are displayed on the Field Definition screen. At runtime, that variable map field displays and allows users to input department names.
  If you were to create a map sectionly, you would have to specify the element names on this screen at this time.
  For further information, see *CA IDMS Mapping Facility Section*.
  A special system field is a field that has a reserved use in a CA ADS application.
  For example, $MESSAGE is a special system field that contains messages (such as error messages) at runtime. A variable field associated with $MESSAGE can display those messages to the user.

- **Edit Picture** -- If you autopaint this map, the length of the field (as defined in the dictionary) will be displayed here. If you create this map sectionly, the length of the field (as defined in the dictionary) will be displayed here after you press [Enter].
  You can create an external picture used for display by entering a different value in Edit picture. For example, you might want to change 9(4).99 to $9(4).99 for a monetary value for display purposes.

- **Display intensity** -- You can specify the runtime display intensity for the variable map field.
  The default, 1 (Normal), causes the field to be displayed at normal intensity. You specify 2 (Bright) to make a field display at bright intensity or 3 (Hidden) to make a field invisible to the user.

- **At end of field** -- You can specify whether the user is restricted from typing beyond the end of the variable map field.
  The default, 1 (Auto-tab), specifies that the field is explicitly delimited. In this case, the user cannot type beyond the end of the field. The cursor will skip to the next unprotected field when the user fills the current field with characters.
  2 (Lock keyboard) causes the keyboard to lock when the user attempts to enter data beyond the end of the field.
  3 (Take no action) specifies that the field is not explicitly delimited. In this case, the user can type beyond the end of the field (although excess characters are truncated on input).
- **Unprotected (/)** -- You can specify whether the user can enter data into the variable map field. Spacing over the slash (/) indicates that the map field is protected and restricts the user from changing the contents of the field.

- **Required (/)** -- You can specify whether the user must enter data into the variable map field. Entering a nonblank character indicates that data must be entered into the field before the map data will be processed.

- **Automatically edited (/)** -- You can enable the automatic data editing feature of CA ADS. For map XXXMAP, you will enable this feature for numeric fields (such as the field that displays department numbers) to make them readable.

- **Skipped by tab key (/)** -- You can specify that that tab key will not stop on this map field.

For more information on using Field Definition screen prompts, see the *CA IDMS Mapping Facility Section*.

**XXXMAP Field Specifications**

The following table summarizes the specifications that you will make when you edit map fields in this step. You will edit each field definition as indicated in this table.

<table>
<thead>
<tr>
<th>Location of field on map</th>
<th>Purpose of field</th>
<th>Specifications for field</th>
</tr>
</thead>
<tbody>
<tr>
<td>After FUNCTION literal field (in the upper left corner)</td>
<td>Displays the name of the application function being executed at runtime</td>
<td>Protected Bright display</td>
</tr>
<tr>
<td>DEPARTMENT INFORMATION literal field</td>
<td>Displays the title for the screen</td>
<td>Bright display</td>
</tr>
<tr>
<td>After DEPARTMENT ID literal field</td>
<td>Displays a department's unique ID number</td>
<td>Auto-tab Automatically edited</td>
</tr>
<tr>
<td>After NAME literal field</td>
<td>Displays a department's name</td>
<td>Auto-tab Pad character - space1</td>
</tr>
<tr>
<td>After HEAD ID literal field</td>
<td>Displays the ID number for the head of the department</td>
<td>Auto-tab Automatically edited</td>
</tr>
<tr>
<td>Last field on the map (on the bottom right side)</td>
<td>Displays runtime messages to the use</td>
<td>Element name: AGR-MESSAGE Length: 80 bytes</td>
</tr>
</tbody>
</table>

1 The pad character is defined on page 2 of the Field Definition screen.

**Modifying XXXMAP**
Modify the variable field shown on the Field Definition screen (AGR-CURRENT-FUNCTION).

⚠️ **Note:** The FUNCTION literal field will not be available for modification because you did not select it on the Layout screen.

Notice that:

- The record element (AGR-CURRENT-FUNCTION) is already displayed for the element name.
- The position of the variable field is shown.
- The edit picture shows the actual length of the function variable field.
- Several defaults are indicated.

To enter the field specifications shown in the preceding table, enter 2 following the **Display intensity** prompt to indicate bright. Space over the slash following **Unprotected** to make this a protected field.

```
Map name: XXXMAP    Version: 1
;FUNCTION: ;________
.;5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row 1 column 1 Drop field (/) _
Element name: AGR-CURRENT-FUNCTION   Subscript
In record ADSO-APPLICATION-GLOBAL-RECORD   Version 1
Edit Picture X(8)
   Display intensity 2 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action
    Unprotected (/) . . . . Required (/) . . . . _
   Automatically edited (/) / Skipped by tab key (/) _
```

DC366004 Specify the variable field and any attributes
F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd

After you press [Enter], MAPC redisplay the Field Definition screen with a message:

- **Map options processed successfully** is returned when the definition contains no errors.
- **An error message** is returned when MAPC detects an error in your definition. In this case, read the message to determine the problem. You can type over any errors and press [Enter] again.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

**Step 8 Edit literal fields**

The next field selected on the Layout screen was the title, DEPARTMENT INFORMATION. When you press [PF5] from the previous Field Definition screen, the **Literal Definition** screen is displayed.
In this step, you will edit a literal field’s definition to determine what the field will look like at runtime. You access the Literal Definition screen from the Layout screen by pressing [PF5] after you have selected fields from the screen, or by pressing [PF5] from another definition screen (either Field or Literal Definition).

There are two pages of data field screens. Navigate between these screens using [PF7] or [PF8], or move directly to the desired page by overtyping the page number.

**Sample Literal Definition Screen - page 1**

Map name: XXXMAP Version: 1

DEPARTMENT INFORMATION

Literal at row 4 column 24 Drop literal (/) _

Display intensity 1 1. Normal 2. Bright 3. Hidden


Outline options (/) . . . . _ Top _ Bottom _ Left _ Right

Sensitive to light pen (/) _

DC366505 Select literal field attributes

F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

Notice that each definition screen shows the field to be defined plus two scales to help you position the field.

You use the Literal Definition screen to edit one map field at a time. After you edit a field (using [PF8] if you need to go to the second page), press PR5. MAPC displays either the Field Definition screen or the Literal Definition screen for the next field that you’ve selected.

When you’ve edited all selected fields, MAPC returns you to the Main Menu screen.

**Note:** Pressing [Enter] on any screen causes it to redisplay so that you can make additional changes if necessary.

**Prompts**

In this step, you will use the following prompts on page 1 the Literal Definition screen:

- **Display intensity** -- You can specify the runtime display intensity for the literal map field. The default, 1 (Normal), causes the field to be displayed at normal intensity. You specify 2 (Bright) to make a field display at bright intensity or 3 (Hidden) to make a field invisible to the user.

For more information on using Literal Definition screen prompts, see the *CA IDMS Mapping Facility Section*. 
Modifying XXXMAP

Modify the literal field shown on the Literal Definition screen.

### Literal Definition

<table>
<thead>
<tr>
<th>Map name: XXXMAP</th>
<th>Version: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80</td>
<td></td>
</tr>
</tbody>
</table>

#### DEPARTMENT INFORMATION

<table>
<thead>
<tr>
<th>Literal at row</th>
<th>4</th>
<th>column</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop literal (//)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Display intensity**
  1. Normal
  2. Bright
  3. Hidden

- **Highlighting**
  1. Blink
  2. Reverse video
  3. Underline

- **Color**
  1. White
  2. Red
  3. Blue
  4. Pink
  5. Yellow
  6. Turquoise
  7. Device default

- **Outline options**
  1. Top
  2. Bottom
  3. Left
  4. Right

- **Sensitive to light pen**
  1. Yes
  2. No

DC366501 Map options processed successfully

F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

After you press [Enter], MAPC redisplay the Literal Definition screen with a message:

- **Map options process successfully** is returned when the definition contains no errors.

An error message is returned when MAPC detects an error in your definition. In this case, read the message to determine the problem. You can type over any errors and press [Enter] again.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

### Defining the Variable Field, DEPT-ID-0410

The next variable field according to the previous table, DEPT-ID-0410, is automatically edited and associated with the auto-tab attribute. These attributes are defaults, and no change needs to be made to this variable field definition on the Field Definition screen.

### Defining the Variable Field, DEPT-NAME-0410

Edit the next variable field according to the previous table. This variable fields requires the auto-tab attribute and a pad character.

**Note:** The pad character is specified on page 2 of the Field Definition screen.
Field at row 7 column 32 Drop field (/) _

Element name: DEPT-NAME-0410 Subscript
In record DEPARTMENT Version 1

Edit Picture X(45)

Display intensity 1 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action

Unprotected (/) _ _ _ _ Required (/) _ _ _ _
Automatically edited (/) _ _ _ _ Skipped by tab key (/) _ _

DC366004 Specify the variable field and any attributes
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

Auto-tab is the default.

To define a pad character, you must go to the second page of the Field Definition screen by pressing [PF8].

Sample Field Definition Screen -- Page 2

Map name: XXXMAP Version: 1

Element name DEPT-NAME-0410 Subscript
In record DEPARTMENT Version 1

Map Read
Transmit data entry (/) _ _ _ _ _ _ _ _
Zero when null (/) _ _ _ _ _ _ _ _
Translate to upper case (/) _ _ _ _ _ _ _ _
Justify data _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ 1 1. Left 2. Right
Pad character format Display _ _ _ _ Hexadecimal _ 40

Map Write
Blank when zero (/) _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Underscore blank fields (/) _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Display without trailing blanks _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Set modified data tag (/) _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Transmit _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ 1 1. Data and attribute byte 3. Erase field 2. Attribute byte only 4. Nothing

DC366404 Select input/output edit options
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F7=Bkwd F8=Fwd

Screen Prompts

- **Pad character format** -- You can identify a pad character for a map field.

For map XXXMAP, you will assign a pad character to the field that displays department names. Entering "40" (the hexadecimal equivalent of a blank) next to **Hexadecimal** for this field ensures that remaining characters are not stored if the user replaces a long department name (for example, SYSTEMS ENGINEERING DEPARTMENT) with a shorter name (for example, SYSTEMS GROUP) and then clears the rest of the field by pressing the ERASE EOF key.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

**Defining the Variable Field, DEPT-HEAD-ID-0410**

You can edit the next variable field, DEPT-HEAD-ID-0410, according to the previous table:
Field Definition  Page 1 of 7
Map name: XXXMAP  Version: 1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
;HEAD ID..: ;____
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row 7 column 32 Drop field (/) _
Element name: DEPT-HEAD-ID-0410  Subscript
In record DEPARTMENT  Version 1
Edit Picture 9(4)
Display intensity 1 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action
Unprotected (/) . . . . . Required (/) . . . . .
Automatically edited (/) / Skipped by tab key (/) _

DC366004 Specify the variable field and any attributes
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

The default is yes for Automatically edited and Auto-tab for At end of field.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

Modifying the Length of the Message Field

You can edit the next variable field, the message field, according to the previous table:

Field Definition  Page 1 of 7
Map name: XXXMAP  Version: 1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
;____
Field at row 21 column 79 Drop field (/) _
Element name: AGR-MESSAGE  Subscript
In record ADSO-APPLICATION-GLOBAL-RECORD  Version 1
Edit Picture x(80)
Display intensity 1 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action
Unprotected (/) . . . . . Required (/) . . . . .
Automatically edited (/) / Skipped by tab key (/) _

DC366004 Specify the variable field and any attributes
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

Change the length to 80. The Edit picture is automatically updated when you press [Enter].

When you are finished editing fields, you can compile the map, as shown in the next step.

Press [PF5] to return to the Main Menu so that you can compile the map.
Step 9 Compile the map

When you compile a map, MAPC creates a load module that incorporates all of your specifications. You compile an map by selecting the Compile activity from the action bar on the Main Menu screen.

You compile the XXXMAP map as shown:

**Compiling the Map**

```
                           Map Compiler
Add       Modify  Compile  Delete  Display  Switch
                        -----------------------------------------
1  1. Compile            Map Compiler
2. View messages
----------------------- International, Inc.
F3=Exit
                        -----------------------------------------
```

- Map name: XXXMAP
- Map version: 1
- Dictionary name: DEMO
- Dictionary node: __________
- Screen: 1. General options
  2. Map-Level help text definition
  3. Associated records
  4. Layout
  5. Field definition

Command ===>
Enter F1=Help F3=Exit F10=Action

To compile the application, position the cursor on the Compile item on the action bar and press [Enter]. You can position the cursor on Compile by:

- Tabbing to Compile and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to Compile and pressing [Enter]
- Typing compile on the command line and pressing [Enter]

Once you have displayed the Compile action item, press [Enter] to compile the map.

After you press [Enter] to compile the map, MAPC displays messages to indicate whether the map has been compiled successfully. You will receive a confirming message if the map has compiled successfully.

An error message is displayed when the application cannot be compiled because of an error. In this case, you can display diagnostic information by selecting the View messages option on the Compile activity on the action bar. Based on this information, you can correct the map and then try to compile the application again.
Exit from MAPC

You can return directly to DC/UCF by pressing [PF3]. Alternatively, you can use the **Switch** activity on the action bar of the Main Menu screen to transfer to another development tool.

In this sample session, you'll exit to DC/UCF so you can execute your application structure. Press [PF3] to exit.

⚠️ **Note**: If you suspend MAPC successfully compiling the current map definition, MAPC saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the map definition. To enable them to access the definition, specify the **Release** option from the **Modify** activity on the action bar on the Main Menu.

After you exit from MAPC, you can display your map as described below.

Optionally display the map

Map **XXXMAP** will be displayed at runtime by dialogs that you define in the next section (Section10, "Defining Dialogs Using ADSC"). To display the map before defining any dialogs, do either of the following:

- In **MAPC**, you can display the map on the MAPC Map Image screen. You will do this later in this section, in Section11, "Modifying a Map Using MAPC”.

- From **DC/UCF**, where you are now, you can display the map by issuing a SHOWMAP command.

Both of the above methods allow you to see how the map will look to an user at runtime. For example, start-field characters are not displayed on the screen; bright fields are displayed in bright intensity on the screen.

Since you have exited from MAPC, you can display map **XXXMAP** directly from DC/UCF:

```
showmap xxxmap
```

**FUNCTION:**

```
DEPARTMENT INFORMATION
```

```
DEPARTMENT ID .......
NAME .......
HEAD ID ..
```

**NEXT RESPONSE:**

Testing the Map

While displaying the map, test out how convenient the map is to use. For example:
Try typing data into unprotected variable fields
Variable fields on the displayed map do not display or store real data.

Try pressing the tab key to advance the cursor from field to field.
When you press [Enter], you return to the DC/UCF display.

Summary 1

In this section, you defined a screen, or map, by using MAPC. XXXMAP contains two types of fields:

- **Literal fields** -- At runtime, literal fields display literal strings.
- **Variable fields** -- At runtime, variable fields display stored values and allow users to input values.

You defined the layout of fields on the map:

1. **You named the records** that would appear on the map.
2. **You named the elements** of those records that would be displayed.
3. **You used the autopaint facility of MAPC** to create a map automatically.
4. **You modified the placement of the elements and added further literal and variable fields.**
5. **You edited fields** by using the Field Definition and Literal Definition screens. You associated each map variable field, not already associated, with a record element or special system field. You also provided additional field characteristics, such as a pad character.

A map can be used by any number of dialogs. For example, XXXMAP is used by dialogs XXXDADD and XXXDUPD in the sample Department application. You will define these dialogs in Section9.

Defining Dialogs Using ADSC

You define dialogs by using the CA ADS dialog compiler (ADSC). The XXXDADD and XXXDUPD dialogs defined in this section are intended for use in the Department application introduced in Section7, "Overview of CA ADS Application Development".

This section includes:

- An overview of developing dialogs for CA ADS applications
- Instructions for defining the sample XXXDADD and XXXDUPD dialogs
- Instructions for executing the sample Department application
- A summary of what you've accomplished in this section
To complete the prototype Department application, you need to define dialogs for the ADDDEP, MODDEP, and DELDEP dialog functions that you created in Section 8, "Defining an Application Structure Using ADSA". When you created ADDDEP, you named XXXDADD as the associated dialog. When you created the MODDEP and DELDEP dialog functions, you named XXXDUPD as the associated dialog for each function. The XXXDADD and XXXDUPD dialogs did not exist when you named them in Section 8.

In this section, you will define the dialogs XXXDADD and XXXDUPD. The diagram below shows how XXXDADD and XXXDUPD fit into the Department application. Function ADDDEP invokes dialog XXXDADD at runtime. Both MODDEP and DELDEP invoke dialog XXXDUPD.

**Note:** You can substitute your initials for XXX in the dialog names.
A **dialog** is an executable module that consists of components defined by using other development tools. For example, a dialog can include a map defined using MAPC, and modules of process code defined using the IDD menu facility.

At runtime, a dialog:

1. Displays a screen to a user
2. Retrieves entries made by the user

**Dialogs in the Department Application**

For example, in the Department application:

- **Dialog XXXDADD** displays a screen that allows users to add department records to the database.
- **Dialog XXXDUPD** displays a screen that allows the user to modify or delete existing department records.

Process code can be executed both before the dialog's screen is displayed and after user input is retrieved.

**Runtime Execution**

Dialogs are executed at runtime whenever control passes to the dialog functions with which they are associated. For example, you associated dialog XXXDADD with the ADDDEP dialog function when you defined ADDDEP in Dialog functions (see page 201). At runtime, dialog XXXDADD will be executed whenever a user invokes the ADDDEP function.

**Note:** It is possible to create **mapless dialogs** that consist only of process logic. For example, a mapless dialog can be defined to perform database operations. You will not create any mapless dialogs for the sample Department application.

**Dialog Components**

As shown in the diagram, a dialog can consist of several components. When you are developing dialogs for a prototype application, it is only necessary to include maps in the dialogs. Defining basic, skeleton dialogs for a prototype application allows users to execute the application and review the screen displays. Later in the application development cycle, you define process modules that perform processing for the dialogs.
IDMSDB--Defining Dialogs Using ADSC (2)

In this section, you will define skeleton dialogs by using the **CA ADS dialog compiler (ADSC)**. A typical ADSC screen is shown below:

```
Add Modify Compile Delete Display Switch
```

```
CA ADS Online Dialog Compiler
CA, Inc.
```

```
Dialog name . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Dialog version . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Dictionary name . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Dictionary node . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Screen . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Screen options:
1. General options
   2. Assign maps
   3. Assign database
   4. Assign records and tables
   5. Assign process modules

```
Copyright (C) 2003 CA, Inc.
```

Command ===>
Enter F1=Help  F3=Exit  F10=Action
Define Dialogs

You use ADSC to define dialogs. To define a skeleton dialog, you invoke ADSC, specify basic information on the dialog, and then create a load module for the dialog.

In this section, you will define skeleton versions of dialogs XXXDADD and XXX DUPD by using ADSC, as follows:

- Step 1 Invoke ADSC (see page 243)
- Step 2 Define Dialog XXXDADD (see page 244)
- Step 3 Name the Associated Map (see page 245)
- Step 4 Create the XXXDADD Dialog Load Module (see page 247)
- Step 5 Define and Compile Dialog XXXDUPD (see page 248)
- Exit from ADSC (see page 249)

After you compile dialogs XXXDADD and XXXDUPD, you can exit from ADSC.

Dialogs for the Department Application

The following table lists specifications for defining skeleton dialogs XXXDADD and XXXDUPD. If you need additional information at any time about the use of ADSC, see .

<table>
<thead>
<tr>
<th>Dialog name</th>
<th>Associated map</th>
<th>Purpose of dialog</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXDADD</td>
<td>XXXMAP1</td>
<td>Allows a user to add a new department record. XXXDADD is invoked by the ADDDEP dialog function.</td>
</tr>
<tr>
<td>xxxdupd</td>
<td>xxxmap1</td>
<td>Allows a user to modify or delete an existing department record. XXXDADD is invoked by the MODDEP and DELDEP dialog functions.</td>
</tr>
</tbody>
</table>

1 You defined map XXXMAP earlier in this sample application development session

Step 1 Invoke ADSC

You can invoke ADSC from CA IDMS/DC or DC/UCF. by specifying the task code for ADSC (for example, ADSCT) in response to the prompt presented by DC/UCF. For example, you can invoke ADSC from CA IDMS/DC as shown:

ENTER NEXT TASK CODE:
adsct

Press the ENTER key to input the - [Enter]
task code for
ADSC.

For more information on task codes for CA ADS development tools, see Application Development Tools (see page 180). ADSC begins by displaying the Main Menu screen. You define a dialog by using the Main Menu screen as described below.
Step 2 Define Dialog XXXDADD

You use the **Main Menu** screen to specify basic information on a dialog. A sample Main Menu screen is shown below:

**Sample Main Menu Screen**

```
Add  Modify  Compile  Delete  Display  Switch

CA ADS Online Dialog Compiler

CA, Inc.

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

Copyright (C) 2003 CA, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

**Screen Prompts**

When you define a skeleton dialog, you typically enter information after one or more of the following Main Menu screen prompts:

- **Dialog name** -- You must specify the same dialog name that you specified when you used ADSA to define the associated dialog function.
  For example, when you define dialog XXXDADD, you must use the same name that you used when you defined the associated ADDDEP function in Section 8, "Defining an Application Structure Using ADSA".

- **Dialog version** -- You must specify a version number, in the range 1 through 9999. The default version is 1.

- **Dictionary name** -- You must specify the same dictionary (if any) as you specified for your application definition in Section 8, "Defining an Application Structure Using ADSA". The correct dictionary name may already be displayed.

- **Dictionary node** -- You must specify the same dictionary node (if any) as you specified for your application definition. The correct dictionary node may already be displayed.

**Defining XXXDADD Dialog**

You can define the XXXDADD dialog on the Main Menu screen:

```
Add  Modify  Compile  Delete  Display  Switch
```

Adding the Dialog

To add the dialog, position the cursor on the **Add** item on the action bar and press ENTER. You can position the cursor on **Add** by:

- Tabbing to **Add** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Add** and pressing [Enter]
- Typing **add** on the command line and pressing [Enter]

Once you have displayed the **Add** action item, press [Enter] to add the dialog to the dictionary. After you press [Enter], the action is confirmed. If there is an error, an error message is displayed.

Step 3 Name the Associated Map

After you specify the name of the dialog, you can name the associated map on the **Map Specifications** screen. You reach the Map Specifications screen by entering 2 next to **Screen** on the Main Menu screen.
Dialog name . . . . . .  XXXDADD
Dialog version . . . . .  1
Dictionary name . . . . .  DEMO
Dictionary node . . . . .  __________

Screen . . . . . . . . .  2  1. General options
  2. Assign maps
  3. Assign database
  4. Assign records and tables
  5. Assign process modules

Copyright (C) 2003 CA, Inc.

Enter  F1=Help  F3=Exit  F10=Action

The following screen is displayed.

Sample Map Specifications Screen

Map Specifications

Dialog  XXXDADD  Version  1

Map name . . . .  ________  Input map . . . . . .  ________
Version . . . .  _____  Version . . . . . .  _____
Label . . . . . .  ________  Label . . . . . .  ________
Paging options  _ 1. Wait
  _ 2. No Wait
  _ 3. Return
Paging mode . . .  Update
  _  Backpage
  _  Auto display
  Output map . . . . . .  ________
  Version . . . . . .  _____
  Label . . . . . .  ________
  Suspension file label  ________

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

Screen Prompts

- **Map name** -- You must specify the name of a map created using MAPC. (See Section 9, "Defining a Screen Display Using MAPC").

- **Map version** -- You must specify an existing version of this map.

Associating a Map with the Dialog

Map Specifications

Dialog  XXXDADD  Version  1

Map name . . . .  XXXXMAP  Input map . . . . . .  ________
Version . . . .  1  Version . . . . . .  _____
Label . . . . . .  ________  Label . . . . . .  ________
Paging options  _ 1. Wait
  _ 2. No Wait
  _ 3. Return
Paging mode . . .  Update
  _  Backpage
  _  Auto display
  Output map . . . . . .  ________
  Version . . . . . .  _____
  Label . . . . . .  ________
Step 4 Create the XXXDADD Dialog Load Module

When you compile a dialog, ADSC creates a load module that incorporates all of your specifications. You compile a dialog by selecting the **Compile** activity from the action bar on the Main Menu screen.

**Compiling the Dialog**

You compile the XXXDADD dialog as shown:

```
Add  Modify  Compile  Delete  Display  Switch
```

```
1 1. Compile  log Compiler
2. Display messages
------------------------- international, Inc.
F3=Exit
```

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

Command ===>
Enter F1=Help  F3=Exit  F10=Action

To compile the application, position the cursor on the **Compile** item on the action bar and press ENTER. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]

Once you have displayed the **Compile** action item, press [Enter] to compile the dialog.

After you press [Enter] to compile the dialog, ADSC displays a message to indicate whether the dialog was compiled successfully.
A confirming message is displayed on the Main Menu screen if the dialog was compiled successfully.

If the dialog could not be compiled, a different message, depending on the nature of the error condition, is displayed. In this case, read the message to determine the problem. After correcting all indicated problems, compile the dialog again.

After you successfully compile the XXXDADD dialog, you can define and compile dialog XXXDUPD as described below.

**Step 5 Define and Compile Dialog XXXDUPD**

To define XXXDUPD, you use the **Main Menu screen**. If you have just compiled dialog XXXDADD as described in Step 4, you can define dialog XXXDUPD on the Main Menu screen by typing over the dialog name on the screen.

The Department application structure you defined earlier associates dialog XXXDUPD with both dialog functions MODDEP and DELDEP. When you define XXXDUPD, you must use the same name that you used defining functions MODDEP and DELDEP.

You use the Main Menu screen to define dialog XXXDUPD:

**Name the Dialog**

Add Modify Compile Delete Display Switch

Name the Dialog Associated with the Dialog

Go to the **Map Specifications** screen and name the map. The map is the same map as that associated with the XXXDADD dialog, XXXMAP.

**Name the Map Associated with the Dialog**

Go to the **Map Specifications** screen and name the map. The map is the same map as that associated with the XXXDADD dialog, XXXMAP.
Create the Load Module

After you define the XXXDUPD dialog, you can create a load module for the dialog. Return to the Main Menu by pressing [PF3] and select the Compile activity.

Command ===>
F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Switch

After you successfully compile the XXXDUPD dialog, you can exit from ADSC or use the Switch activity to access another tool.

Exit from ADSC

In this sample session, you’ll exit to DC/UCF so you can execute your application structure again. Press [PF3] to exit from the Main Menu screen to exit.
After you exit from ADSC, you can execute your application again as described below.

Test the Application

Once you define dialogs for functions ADDDEP, MODDEP, and DELDEP, all menu, system, and dialog functions in the sample Department application are executable.

You now can fully test the prototype Department application, as follows:

- Step 1 Invoke the Application (see page 250)
- Step 2 Test Features of the Prototype (see page 251)
- Step 3 Exit from the Application (see page 255)

Step 1 Invoke the Application

You execute a CA ADS application under the CA ADS runtime system. When you executed the partially defined Department application in Section 8, "Defining an Application Structure Using ADSA", you invoked the application by entering two task codes. The first task code (ADS) invoked the runtime system; the second (XXXDEPT) invoked the application itself.

Dynamically Associating the Task Code

To make it easier to execute the application, you can dynamically associate the application's task code with the runtime system. To do this, you issue a DCMT VARY DYNAMIC TASK command while using DC/UCF.

For example, you can enter this DCMT command while using CA IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
dcmt vary dynamic task xxxdept invokes adsort run1.
```

ADSORUN1 is the internal name for the CA ADS runtime system. After you issue the above DCMT command, entering task code XXXDEPT invokes the runtime system and then causes the Department application to be executed. This association remains in effect until the system is recycled.

Invoke the Application

You can now invoke the Department application from DC/UCF by entering the task code for the application:
The Department application's task code has been associated with the runtime system.

DEPTMENU Screen

The DEPTMENU screen is the first screen displayed.

RESPONSE: SEND DATA- MODE: STEP

After you invoke the Department application from DC/UCF, you can test out features that you've implemented in the prototype Department application, as described below.

Step 2 Test Features of the Prototype

The first function executed in the sample application is DEPTMENU. According to your application design, the following responses are valid from DEPTMENU:

- **ADD** selects dialog function ADDDEP. In the final application, ADDDEP will allow users to add new department information in the database.

- **MOD** selects dialog function MODDEP, which will allow users to modify existing department information in the database.

- **DEL** selects dialog function DELDEP, which will allow users to delete department information.

- **EXIT** selects system function QUIT, which will allow users to leave the application.

Test out each of the above responses while you are executing the Department application prototype. For example, try specifying the ADD response:

**Specifying the ADD Response**

To specify the ADD response, you can use any of the following methods:

- Press [PF4] to select the ADD response

- Type a nonblank character in front of a response and press [Enter] to select the response.
ADDDEP Function

The ADDDEP function is displayed with the XXXMAP.

FUNCTION: ADDDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........:
NAME .........:
HEAD ID ..:

NEXT RESPONSE:

The ADDDEP function allows the user to enter information on a department. Try tabbing between variable fields and entering sample department information. You cannot type anything in the FUNCTION: field because you made it a protected field.

FUNCTION: ADDDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 9012
NAME .........: Application Testing
HEAD ID ..: 3456

NEXT RESPONSE:

ADDDEP is a skeleton dialog, which means that you haven’t added any process logic to the dialog. The dialog cannot access the database. Therefore, your sample data is not stored when you press [Enter].

Input-Handling Operations

Other input-handling operations are performed automatically at runtime. For example, your sample input is:
- **Echoed on the screen** after you press [Enter].

- **Tested for invalid values** and redisplayed in bold when errors are found.
  
  For example, try entering invalid values in the DEPARTMENT ID and HEAD ID variable fields:

  ```
  FUNCTION: ADDDEP

  DEPARTMENT INFORMATION

  DEPARTMENT ID .......: xyz
  NAME ......: Quality Assurance
  HEAD ID ..: xyz

  RESPONSE:
  ```

  In this case, the runtime system returns an error message:

  ```
  **ERROR AT 7,24** **ERROR AT 9,24**
  ```

  Nonnumeric data is invalid for the DEPARTMENT ID and HEAD ID variable fields because you enabled the CA ADS automatic editing feature for these numeric fields in Section 8.

  For more information on how automatic editing can be used to keep users from entering invalid values, see the *CA IDMS Mapping Facility Section*.

  You also can test the NEXT RESPONSE variable field. You can try entering undefined responses (for example, UPDATE) or responses that are valid for the application but not for the ADDDEP function (for example, MOD). In fact, a user testing the Department application prototype probably would try to access the MODDEP function from ADDDEP to see if newly added department information can be modified easily if, for example, the department name is misspelled.

  Enter new department values on the screen, and then specify the MOD response to try accessing MODDEP from ADDDEP:

  **Specifying the MOD Response from ADDDEP**

  ```
  FUNCTION: ADDDEP

  DEPARTMENT INFORMATION

  DEPARTMENT ID .......: 4567
  NAME ......: System Software Division
  HEAD ID ..: 9521

  NEXT RESPONSE: mod
  FUNCTION: ADDDEP

  DEPARTMENT INFORMATION

  DEPARTMENT ID .......: 4567
  NAME ......: SYSTEM SOFTWARE DIVISION
  HEAD ID ..: 9521
  ```
RESPONSE:

DC172008 *** UNACCEPTABLE RESPONSE. PLEASE TRY AGAIN ***

Even though MOD is defined for the application, it is not valid from the ADDDEP function. According to your application definition, only the following responses are valid from ADDDEP:

- **BACK** selects system function POP, which returns execution to the previous menu function (in this case, DEPTMENU).
- **EXIT** selects system function QUIT, which terminates the application.

**Display the MODDEP Function**

To display the MODDEP function from ADDDEP, you must first access a function from which MODDEP is valid. Since MOD is valid for the DEPTMENU function, use the **BACK** response to return to DEPTMENU, and then invoke MOD from DEPTMENU:

```plaintext
FUNCTION: ADDDEP

DEPARTMENT INFORMATION

DEPARTMENT ID .........: 4567
    NAME ......: SYSTEM SOFTWARE DIVISION
    HEAD ID ..: 9521

RESPONSE: back
```

Choose MOD from the DEPTMENU screen to display the MODDEP function.

```
DIALOG:          PAGE: 1 OF: 1
DATE: 08/19/99   NEXT PAGE:

DEPARTMENT INFORMATION APPLICATION
MAIN MENU

  ADD (PF4)      ADD A NEW DEPARTMENT
  MOD (PF5)      MODIFY A DEPARTMENT
  DEL (PF6)      DELETE A DEPARTMENT
  EXIT (PF9)     TERMINATE APPLICATION
```

The MODDEP function is displayed

```
FUNCTION: MODDEP

DEPARTMENT INFORMATION
```

**Display the DELDEP Function**

To display function DELDEP, you first return to the DEPTMENU function. From DEPTMENU, you can invoke DELDEP as shown:
CA IDMS - 19.0

FUNCTION: MODDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 0000
NAME .......:
HEAD ID ..: 0000

RESPONSE: back

Select the DEL response to display DELDEP

DIALOG: DATE: 08/19/99 PAGE: 1 OF: 1 
DEPARTMENT INFORMATION APPLICATION NEXT PAGE:
MAIN MENU

_ ADD (PF4) ADD A NEW DEPARTMENT
_ MOD (PF5) MODIFY A DEPARTMENT
x DEL (PF6) DELETE A DEPARTMENT
_ EXIT (PF9) TERMINATE APPLICATION

The DELDEP function is displayed.

FUNCTION: DELDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 0000
NAME .......:
HEAD ID ..: 0000

RESPONSE:

Continue to test the application prototype until you are familiar with the Department application prototype. When you are finished testing the application, exit from the application as described below.

Step 3 Exit from the Application

When you are ready to exit from the application, select the EXIT response. The EXIT response invokes the QUIT system response, which terminates the application and returns control to DC/UCF.

When you defined the Department application structure, you defined EXIT to be available from all functions in the application. You can exit from the Department application by selecting EXIT from any function and pressing the [Enter] key. For example, while using the DELDEP function, you can exit from the Department application as shown:

FUNCTION: DELDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 0000
NAME .......:
HEAD ID ..: 0000
RESPONSE: exit

You can also press [PF9] the control key associated with EXIT.

If you want to execute the prototype again, specify the application’s task code to DC/UCF and press [Enter], as described above in “Step 1: Invoke the Application”.

Summary 2

Developers and users can execute a preliminary (that is, prototype) application early in the development cycle. The prototype can be used to test the user interface of the application and to provide a milestone in the application development cycle. In this section, you completed and executed the prototype of the Department application. By executing the prototype, you were able see how a user might use the application.

You created the Department application prototype by defining application components as described below:

1. You defined the application structure as described in Defining an Application Structure Using ADSA (see page 182). The application structure consists of:
   - Functions, which represent the units of work to be performed by the application. You defined menu, system, and dialog functions for the Department application.
   - Responses, which establish runtime paths between the application functions.
   - Task codes, which establish entry points into the application.

2. You defined a map, or screen display, as described in Defining a Screen Display Using MAPC (see page 213). The map you created will be displayed by the dialogs in the Department application.

3. You defined skeleton dialogs for the ADDDEP, MODDEP, and DELDEP dialog functions in this section. Defining skeleton dialogs allowed you to execute the application and test flow of control and screen displays. In later sections, you will create modules of process code for the ADDDEP, MODDEP, and DELDEP dialogs.

Changes to the Application

Based on tests made using the prototype, users and developers often suggest modifications to the application. For example, users who test out the Department application probably would request that a path be defined that leads directly from ADDDEP to the MODDEP function.

Other changes to the application may be suggested to make the application conform to site conventions. For example, one convention is to use the [PF3] to leave an application Both of the above changes to the Department application can be made by using the application compiler (ADSA), as detailed in the next section.
Users also can suggest changes to maps. For example, users might request that key data be displayed in bright intensity and that error messages for variable fields provide specific information. MAPC is used to make changes to maps, as described in Section 11.

Modifying the Application Structure Using ADSA

Developers and end users can execute the prototype of a CA ADS application to review the application's structure and user interface. Based on the prototype, they can suggest changes to the application. This section provides instructions for modifying the Department application structure.

This section includes:

- An overview of modifying an application structure in the CA ADS environment
- Instructions for modifying the sample Department application
- A summary of what you've accomplished in this section

User requests and design alterations can be incorporated easily into a prototype application. Minor changes, such as changing the function key that invokes a response, and major changes, such as adding new responses and functions to the application, can be performed quickly and easily.

You modify application components by using the same tools you use to define the components. To modify the Department application structure, you will use the CA ADS application compiler (ADSA).

In this section, you will use ADSA to:

1. **Modify the EXIT response** so that [PF3] invokes the response
2. **Modify the ADDDEP function** so that the MOD response is valid directly from ADDDEP

The following diagram shows how the modifications affect the structure of the Department application. Using ADSA, you will make the MOD response valid from the ADDDEP function and also change the function key for the EXIT response. Instructions for modifying the sample application and executing the modified application are provided on the following pages.
Steps to Modify Department Application

To modify the Department application, you will perform the following steps:

- Step 1 Retrieve the Application to be Modified (see page 258)
- Step 2 Select Responses and Functions (see page 260)
- Step 3 Modify the EXIT Response (see page 261)
- Step 4 Modify the ADDDEP Function (see page 261)
- Step 5 Recompile the Application (see page 262)

After you modify the application, you can exit from ADSA and execute the application to see how your changes impact runtime flow of control. If you need additional information at any time about the use of ADSA, see Defining an Application Structure Using ADSA.

Step 1 Retrieve the Application to be Modified

Invoking ADSA

In order to modify an application, you must first invoke ADSA and then use ADSA to retrieve the application definition.
You invoke ADSA from CA IDMS/DC or DC/UCF by specifying the task code for ADSA (for example, ADSAT) in response to the prompt presented by DC/UCF. For example, when using CA IDMS/DC, you invoke ADSA as shown:

ENTER NEXT TASK CODE:
adsat

[Enter]

For more information on invoking ADSA, see Step 1: Invoke ADSA (see page 189).

ADSA begins by displaying the Main Menu screen. You use the ADSA Main Menu screen to retrieve an application definition for modification. To retrieve an application, you typically enter information after one or more of the following Main Menu screen prompts:

**Screen prompts**

- **Application name** -- You must specify the name (for example, XXXDEPT) that you used when you defined the application.

- **Dictionary name** -- You must specify the same dictionary, if any, as you specified for your application definition. The correct dictionary name may already be displayed in this field.

- **Dictionary node** -- You must specify the same dictionary node, if any, as you specified for your application definition. The correct dictionary node may already be displayed in this field.

**Retrieving the Department application**

Use the ADSA Main Menu screen to retrieve the Department application:

```
Add  Modify  Compile  Delete  Display  Switch
```

[ CA ADS Application Compiler ]

CA, Inc.

```
Application name . . .    xxxappl
Application version . .  1
Dictionary name . . .    demo
Dictionary node . . .    __________
Screen . . . . . . . . .  1. General options  
                        2. Responses and Functions 
                        3. Global records  
                        4. Task codes  
```

After you press [Enter], ADSA redisplay the Main Menu screen with a message confirming that the application is available for modification.
Note: If the application has not been explicitly released (using the Release option of the Modify action on the action bar of the Main Menu), naming the application on the Main Menu screen retrieves that definition for modification. If the application has been released, you use the ADSA Main Menu screen to check out the application definition for modification (using the Checkout option of the Modify action on the action bar of the Main Menu).

For information on checking out an application, see CA ADS Reference Section.

If the application has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

If you made any errors in your application specification, ADSA displays information on another application, and/or displays an error message. In either case, make sure that you typed the correct application name, dictionary, and node, as necessary. You can type over any errors, and then press [Enter] again.

After you successfully retrieve the Department application, you can modify the EXIT response.

Step 2 Select Responses and Functions

In this step, you will select the response and the function that you want to modify (EXIT response and ADDDEP function).

Response/Function List screen

Choose 2 at the Screen prompt on the Main Menu screen and press [Enter]. This will bring you to the Response/Function List screen. On the Response/Function List screen, select the response and function you want to change.

```
Response/Function List Page 1 of 1

Application name: XXXAPPL Version: 1

<table>
<thead>
<tr>
<th>Select (/)</th>
<th>Response name</th>
<th>Assigned key</th>
<th>Select (/)</th>
<th>Function name/type(1,2,3)*</th>
<th>Program/Dialog name</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>EXIT</td>
<td>PF09</td>
<td>_</td>
<td>QUIT /</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>ADD</td>
<td>PF01</td>
<td>/</td>
<td>ADDDEP / 1</td>
<td>XXXDADD</td>
</tr>
<tr>
<td>_</td>
<td>MOD</td>
<td>PF02</td>
<td>/</td>
<td>MODDEP / 1</td>
<td>XXXDUPD</td>
</tr>
<tr>
<td>_</td>
<td>DEL</td>
<td>PF03</td>
<td>_</td>
<td>DELDEP / 1</td>
<td>XXXDUPD</td>
</tr>
<tr>
<td>_</td>
<td>BACK</td>
<td>CLEAR</td>
<td>_</td>
<td>POP /</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td></td>
<td></td>
<td>_</td>
<td>DEPTMENU / 3</td>
<td></td>
</tr>
</tbody>
</table>

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

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

Pressing [PF5] will display the Response Definition screen.
Step 3 Modify the EXIT Response

Modifying the EXIT response definition

In this step, you will change the control key for the EXIT response from [PF9] to [PF3]. You use either the Response/Function List screen or the Response Definition screen to modify an application response assigned key. Use the Response Definition screen to modify the EXIT response in the sample application.

Response Definition screen

Press [PF5] to display the Response Definition screen for the EXIT response.

Type the name of the new control key over the previous control key:

Response Definition

<table>
<thead>
<tr>
<th>Application name: XXXAPPL</th>
<th>Version: 1</th>
<th>Drop response (/) _</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response name: EXIT</td>
<td></td>
<td>QUIT</td>
</tr>
<tr>
<td>Function invoked: QUIT</td>
<td></td>
<td>TERMINATE APPLICATION</td>
</tr>
<tr>
<td>Description . . . . . . .</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response type . . . . . .</td>
<td>1. Global</td>
<td>2. Local</td>
</tr>
<tr>
<td>Response execution . . . .</td>
<td>1. Immediate</td>
<td>2. Deferred</td>
</tr>
<tr>
<td>Assigned key . . . . . . .</td>
<td>pf3</td>
<td></td>
</tr>
<tr>
<td>Control command . . . . .</td>
<td>1. Transfer</td>
<td>2. Invoke</td>
</tr>
<tr>
<td></td>
<td>3. Link</td>
<td>4. Return</td>
</tr>
<tr>
<td></td>
<td>5. Return continue</td>
<td>6. Return clear</td>
</tr>
<tr>
<td></td>
<td>7. Return continue clear</td>
<td>8. Transfer nofinish</td>
</tr>
<tr>
<td></td>
<td>9. Invoke nosave</td>
<td>10. Link nosave</td>
</tr>
</tbody>
</table>

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

After you modify the EXIT response and press [Enter], ADSA redisplays the Response Definition screen with a confirming message.

As soon as you successfully modify the EXIT response, you can modify the ADDDEP function. Press [PF5] to access the Function Definition screen for the ADDDEP function.

Step 4 Modify the ADDDEP Function

Make the MOD response valid from ADDDEP by selecting the response on the Function Definition screen.

Function Definition (Dialog)

<table>
<thead>
<tr>
<th>Application name: XXXAPPL</th>
<th>Version: 1</th>
<th>Drop function (/) _</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function name: ADDDEP</td>
<td></td>
<td>ADD DEPARTMENT</td>
</tr>
<tr>
<td>Description . . . . .</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associated dialog . . .</td>
<td>XXXDADD</td>
<td>User exit dialog . . .</td>
</tr>
<tr>
<td>Default response . . . .</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid response(/) Response Key Function Valid response(/) Response Key Function</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Step 5 Recompile the Application

After you modify an application, you must recompile the application to create an updated load module for the application. You recompile the Department application by selecting the **Compile** activity.

**Compiling the application**

To compile the application, position the cursor on the **Compile** item on the action bar and press [Enter]. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing `compile` on the command line and pressing [Enter]

Once you have displayed the **Compile** action item, press [Enter] to compile the application.
After you press [Enter] to compile the application, ADSA displays either a confirming message to indicate the you successfully compiled the application, or an error message when the application cannot be compiled because of an error.

In the case of compilation errors, you can display diagnostic information by selecting the View messages activity from the action bar. Based on this information, you can correct the application and then try to compile the application again.

Exit From ADSA 1

You can also return directly to DC/UCF by pressing [PF3]. Alternatively, the Switch activity on the action bar of the Main Menu screen allows you to exit from ADSA and transfer to another development tool.

In this sample session, you'll exit to DC/UCF so you can execute your application structure. Press [PF3] to exit.

⚠️ Note: If you leave ADSA without successfully compiling the current application definition, ADSA saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the application definition. To enable them to access the definition, specify the Release option from the Modify activity on the action bar on the Main Menu.

After you exit from ADSA, you can execute your application as described below.

Execute the Application

In the previous steps, you made the following changes to the structure of the Department application:

- You assigned [PF3] to the EXIT response.
- You made the MOD response valid from the ADDDEP function.

You can now execute the Department application and see how your changes affect the way that end users of the Department application move from one function to another.

You invoke an application from DC/UCF by entering the task code (XXXDEPT) for the application. For example, from CA IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:  
xxxdept
[Enter]
```

For more information on invoking the Department application, see Instructions For Executing the Application (see page 250).
After you invoke the application, display the ADDDEP function. To test how you transfer from ADDDEP to function MODDEP, you select the MOD response:

FUNCTION: ADDDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 0000
   NAME .......
   HEAD ID ..: 0000

NEXT RESPONSE: mod

The MODDEP function is displayed.

FUNCTION: MODDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........:
   NAME .......
   HEAD ID ..:

Testing the BACK response

According to your application design, both the BACK and EXIT responses are valid from MODDEP. Try requesting BACK:

FUNCTION: MODDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........:
   NAME .......
   HEAD ID ..:

NEXT RESPONSE: back

Notice that the BACK function takes you to DEPTMENU, rather than to ADDDEP. This is because BACK invokes the POP system function, which returns control to the most recently executed menu in the application.

Pressing [PF3] to invoke the EXIT response

When you are ready to leave the application, you can test out the EXIT response. You can test the EXIT response from any function in the Department application. To test EXIT, try pressing [PF3] to select the EXIT response:

FUNCTION: MODDEP

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 0000
   NAME .......
   HEAD ID ..: 0000
CA IDMS - 19.0

PF3 is now associated -- [PF3] with the EXIT response throughout the application.

When you press [PF3], the EXIT response is invoked. The associated system function, QUIT, exits you from the Department application. To test out other features of the Department application, you can invoke the application again, as described earlier.

Summary 3

You can use ADSA to modify the application structure at any time. You can use ADSA to add, modify, and delete functions, responses, and task codes, as necessary.

In this section, you used ADSA as follows:

1. **You modified the EXIT response** to make [PF3] invoke the EXIT response.

2. **You modified the ADDDEP function** to make the MOD response available directly from ADDDEP.

You can modify any part of the Department application structure at this stage. For example:

- If administrators decide that a particular function is no longer necessary due to changes in regulations, you can use ADSA to quickly delete the function and the response that invokes it.

- If end users decide that a summary or list screen would be useful at some point in the application, you can use ADSA to define a new function to display the summary screen. You can then add a response to invoke the new function, and make other responses valid from the new function.

Additionally, you can modify other application components at any time in an application's life cycle. As an application developer, you can make changes to maps, for example, as soon as end user suggestions are approved. In the next section, you will use MAPC to modify map XXXMAP.

Modifying a Map Using MAPC

In the previous section, you modified the structure of the Department application based on preferences at the site. Map layouts can also be modified to satisfy end-user and site requirements. As a developer, you can modify a map's layout as soon as the modifications are suggested and approved.

This section provides instructions for using MAPC to modify the layout of XXXMAP, and includes:

- An overview of modifying maps

- Steps for modifying sample map XXXMAP

- Steps for associating the updated map with dialogs that use the map

- Steps for executing the application
A summary of what you've accomplished in this section

Maps can be modified easily during development or at any other time in an application's life cycle. Modifications can be suggested by development staff and end users. For example, end users who execute the prototype Department application can request that the department ID variable field be displayed in bright intensity to make it easier to locate on the screen.

You can modify a map to make the map conform to screen-display conventions at a given site. For example, it may be necessary to display the current date on each map, or to change the location of particular fields on the map.

Changes to XXXMAP

In this section, you will make changes to map XXXMAP so that:

- **The current date is displayed on the map** -- You will add a variable field to display the current date at runtime, and an adjacent literal field (DATE) to label the displayed information.

- **The department ID number is displayed in bright intensity** -- You will modify the variable field that displays department ids to make the data display in bright intensity at runtime.

- **Error messages for fields provide specific information** -- You will modify definitions for both the department ID and department head ID variable fields to define specific error messages for the fields.

- **The NEXT RESPONSE literal and variable fields are displayed at a higher row on the screen** -- You will modify the NEXT RESPONSE literal and variable fields to move them to row 18 (to make map XXXMAP conform to other nonmenu screens at the site).

The following screen shows the layout of the modified map. To modify map XXXMAP, you will use the **online mapping facility (MAPC)**, which you used to define the map in Section8, "Defining an Application Structure Using ADSA". After you modify map XXXMAP, you need to update dialogs that use the modified map. To do this, you will use the **CA ADS dialog compiler (ADSC)**, which you used to define dialogs in Section10, "Defining Dialogs Using ADSC".

```
FUNCTION: __________
DATE....: __________

DEPARTMENT INFORMATION

DEPARTMENT ID ........: ___
NAME ........: ________________________________
HEAD ID ..: ___

NEXT RESPONSE: __________

-------------------------------------------------------------------------------------------------------------------------------
```

MAPC and ADSC procedures are presented below, followed by a discussion of executing the application.
Modifying a Map Using MAPC

In this procedure, you will use MAPC to modify the layout of map XXXMAP. You will perform the following steps:

1. Retrieve the map to be modified.
2. Add and select map fields.
3. Edit selected fields.
4. Optionally display the map layout.
5. Recompile the map load module.

These steps are described below. Instructions for associating the modified map definition with dialogs that use the map are presented in "Updating Modified Maps In Dialogs Using ADSC", later in this section.

Step 1 Retrieve the Map To Be Modified

In order to modify a map, you must first invoke MAPC.

You can invoke MAPC from CA IDMS/DC or DC/UCF by specifying the task code for MAPC (for example, MAPCT) in response to the prompt presented by DC/UCF. For example, when using CA IDMS/DC, you invoke MAPC as shown:

ENTER NEXT TASK CODE:
mapct
[Enter]

For more information on invoking MAPC, see Step 1: Invoke MAPC (see page ).

Screen Prompts

To retrieve a map, you typically enter information after one or more of the following Map Definition screen prompts:

- **Map name** -- You must specify the name (XXXMAP) that you used when you defined the map in Section 9, "Defining a Screen Display Using MAPC".

- **Dictionary name** -- You must specify the same dictionary, if any, as you specified for your map. The correct dictionary name may already be displayed in this field.

- **Dictionary node** -- You must specify the same dictionary node, if any, as you specified for your map definition in Section 14, "Modifying Process Logic in a Dialog". The correct dictionary node may already be displayed in this field.

Use the MAPC Main Menu screen to retrieve map XXXMAP:
After you press [Enter], MAPC redisplay the Main Menu screen with a message confirming that the map is available for modification.

`Note:` If the map has not been explicitly released (using the `Release` option of the `Modify` action on the action bar of the Main Menu), naming the map on the Main Menu screen retrieves that definition for modification. If the map has been released, you use the MAPC Main Menu screen to check out the map definition for modification (using the `Checkout` option of the `Modify` action on the action bar of the Main Menu).

For information on checking out a map, see `CA IDMS Mapping Facility Section`.

If the map has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

After you successfully check out `XXXMAP`, you can modify the map layout by adding and selecting map fields.

### Step 2 Add and Select Map Fields

When you defined the layout for `XXXMAP` in `Step 5: Modify the map layout (see page)`, you modified the placement of fields on the map by using the Layout screen. In this section, you will use the Layout screen to add new fields to the map layout and to select existing fields for modification.

From the Main Menu screen, proceed to the Layout screen entering `4` for the `Screen` prompt and pressing [Enter]:

Add Modify Compile Delete Display Switch  

CA IDMS/DC Online Map Compiler
Layout Screen

The Layout screen for the map is displayed:

;FUNCTION: ;________

;DEPARTMENT INFORMATION

;DEPARTMENT ID .......: ;____*
;NAME ........: ____________________________________________*
;HEAD ID ..: ;**

;NEXT RESPONSE: ;______*

Note: The example above shows the entire map. On the Layout screen, the bottom of the map is hidden by the list of available function keys. Use [PF8] to see the hidden portion of the map.

When you first display the Layout screen for an existing map, each field on the map is preceded by a start-field character, as shown above. While using the Layout screen, you can select a field for editing by:

- Pressing [PF2] while on the field you want to select
- Typing a select-field character (%) in place of the start-field character for the field.
You use the start-field and select-field characters based on the following sectionlines:

- The **start-field character** (default is ; or () defines the start of a field on the Layout screen.

  ![Note: On the screens shown in this section, ; indicates a start-field character.](image)

- The **select-field character** (default is %) defines the start of a field and simultaneously selects the field for editing.

You use the Layout screen to modify the layout of map XXXMAP and to select fields on the map for further editing:

**Modifying the Map Layout**

Make the indicated specifications:

- Begin the new DATE literal field with a start-field character (shown here as ;)

- Begin the DATE variable field with a select-field character (shown here as ;)

- Type a select-field character (or press [PF2]) over the start-field character for variable fields to select them for editing (DEPARTMENT ID variable field and HEAD ID variable field).

- Select the NEXT RESPONSE literal and variable fields for editing by using the select-field character or [PF2].

  ;FUNCTION: ;________
  ;date.....; 
  ;DEPARTMENT INFORMATION

  ;DEPARTMENT ID .......: %____*
  ;NAME .....; ____________________________________________*
  ;HEAD ID ..: %____*

  %NEXT RESPONSE: %______ *

  __________________________________________________________;

After you press [Enter], MAPC redisplay the Layout screen so that you can inspect the screen. You can correct any errors on the Layout screen in either of the following ways:

- **To change a few fields**, type over the characters that you want to change and press [Enter] again.

- **To erase the modifications that you just made**, press the CLEAR key.

  In this section, pressing CLEAR on the Layout screen does not erase fields that you defined when you originally generated the map in Section9, "Defining a Screen Display Using MAPC".
Step 3 Edit the Selected Fields

To edit new and existing fields in this section, you will use the Field and Literal Definition screens to make specifications for fields, such as the display intensity of the runtime field, define error messages for the department ID and department head ID variable field, and so forth.

You selected fields for editing by using the Layout screen earlier in this section, in Step 2: Add and Select Map Fields (see page 268). The following table summarizes the specifications that you will make when editing each selected field. You will edit each selected field’s definition as indicated in this table.

For more information on using prompts to edit field definitions, see Step 5: Modify the map layout (see page ).

<table>
<thead>
<tr>
<th>Location of field on map</th>
<th>Purpose of field</th>
<th>Specifications for field</th>
</tr>
</thead>
<tbody>
<tr>
<td>After DATE literal field (in the upper right corner)</td>
<td>Displays the current date</td>
<td>Element name: AGR-DATE1 Protected Edit Picture XX/XX/XX</td>
</tr>
<tr>
<td>After DEPARTMENT ID literal field</td>
<td>Displays a department's unique ID number</td>
<td>Bright display Error message: <em>ENTER A NUMERIC DEPT ID</em></td>
</tr>
<tr>
<td>After HEAD ID literal field</td>
<td>Displays the ID number for the head of the department</td>
<td>Error message: <em>ENTER A NUMERIC DEPT ID</em></td>
</tr>
<tr>
<td>Literal NEXT RESPONSE in lower left corner</td>
<td>Prompts the user to input a response name</td>
<td>ROW .......... : 18</td>
</tr>
<tr>
<td>After NEXT RESPONSE literal field</td>
<td>Allows a user to input a response name</td>
<td>ROW .......... : 18</td>
</tr>
</tbody>
</table>

1 AGR-DATE is an element in ADSO-APPLICATION-GLOBAL-RECORD

Press [PFS] from the Layout screen to begin editing fields. This brings you to the first definition screen, a Field Definition screen for the DATE variable field.

Editing the DATE Variable Field

On the Field Definition screen, you edit the DATE variable field information as shown:

- The field being edited is highlighted on the screen.
- Name the record element (in this example, AGR-DATE) to be associated with the variable field.
- Specify an edit picture of **xx/xx/xx**.
DATE.....: _
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row 2 column 16 Drop field (/) _

Element name: agr-date
In record

Edit Picture xx/xx/xx

Display intensity 1 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action

Unprotected (/) . . . . . / Required (/) . . . . . .
Automatically edited (/) / Skipped by tab key (/) _

DC366004 Specify the variable field and any attributes
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

When you press [Enter], the Field Definition screen is redisplayed with a confirming message.

Field Definition Page 1 of 7
Map name: XXXMAP Version: _

FUNCTION: __________
DATE....: __________
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row 2 column 16 Drop field (/) _

Element name: AGR-DATE Subscript
In record ADSO-APPLICATION-GLOBAL-RECORD Version 1

Edit Picture XX/XX/XX

Display intensity 1 1. Normal 2. Bright 3. Hidden
At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action

Unprotected (/) . . . . . / Required (/) . . . . . .
Automatically edited (/) / Skipped by tab key (/) _

DC366001 Map options processed successfully
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

This message informs you that the new field definition contains no errors and has been successfully added to the map. Underscores displayed for the field show you the length of the variable field.

Editing the DEPARTMENT ID Variable Field

The next map field selected for editing is the DEPARTMENT ID variable field. To edit this field, you will:

1. Make data in the field display in bright intensity.

2. Define an error message for the field. To do this, you will use page 3 - Additional Edit Criteria - of the Field Definition screen.

Pressing [PF5] brings the Field Definition screen highlighting the variable field for the department ID.

Field Definition Page 1 of 7
Map name: XXXMAP Version: _
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
DEPARTMENT ID ........: _____
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row 2 column 16 Drop field (/) _

  Element name: DEPT-ID-0410 Subscript
  In record DEPARTMENT Version 100
  Edit Picture 9(4)
  Display intensity 2 1. Normal 2. Bright 3. Hidden
  At end of field 1 1. Auto-tab 2. Lock keyboard 3. Take no action
  Unprotected (/) . . . . . / Required (/) . . . . . _
  Automatically edited (/) / Skipped by tab key (/) _

DC366004 Specify the variable field and any attributes
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F8=Fwd

When you press [Enter] to input your modification, the Field Definition screen displays a confirming
message.

This message informs you that an existing field definition contains no errors and has been successfully modified in the map.

Defining an Error Message

To define a message for this field, press [PF8] twice (or change the page number in the upper right corner) to get to page 3 - Additional Edit Criteria - of the Field Definition screen.

Sample Additional Edit Criteria Screen

Map name: XXXMAP Version: 1
  Additional Edit Criteria Page 3 of 7
  Element name DEPT-ID-0410 Subscript
  In record DEPARTMENT Version 1
  Edit table name . . . _________ Version ___ Link with map (/) _
  Edit type . . . . . . . . 1. Valid values 2. Invalid values
  Code table name . . . _________ Version ___ Link with map (/) _
  Error message (specify ID or text)
  ID. . . . . . . . Prefix ___ Number ______
  Text. . . . . . . . ______________________________________________________
  ______________________________________________________

DC365804 Specify edit options
F1=Help F3=Exit F4=Prev F5=Next F6=Preview F7=Bkwd F8=Fwd

Screen Prompts

Error message -- :i1.ERROR MSG Enter the text for the message that will display for the field when an error is encountered.

Add the error message associated with the DEPARTMENT ID variable field as shown:
A confirming message is displayed when your specifications on the Additional Edit Criteria screen contain no errors and have modified the field definition.

Press [PF5] to have the Field Definition screen for the next select field displayed.

**Editing the HEAD ID Variable Field**

The next field selected for editing is the HEAD ID variable field. You need to define an error message for this field by using the Additional Edit Criteria screen.

When you have successfully defined an error message for the DEPT-HEAD-ID variable field, you can edit the next selected field. To display the next selected field, press [PF5]. MAPC displays the Literal Definition screen for the NEXT RESPONSE literal field.
To move the NEXT RESPONSE literal field, change the row number to reflect the new position. (Remember that you could have used the Layout screen with the alternate PF keys to move both this literal field and its variable field.)

After you press [Enter], the NEXT RESPONSE literal field is displayed in its new location.

Displaying and Moving the NEXT RESPONSE Variable Field

Press [PF5] to see the next field to be defined. The next field is the NEXT RESPONSE variable field. Move this field to row 18.

After you press [Enter], the Field Definition screen for the response variable field is redisplayed with a confirming message.

You have edited all of the fields you selected for editing when you used the Layout screen earlier in this section, in Step 2: Add and Select Map Fields (see page 268).

Pressing [PF5] brings you to Main Menu screen.

Before you recompile the map, you can display the current layout for map XXXMAP.

Step 4 Optionally Display the Map Layout

You can use the Map Image screen in MAPC to see how the modified map will look to a user. This allows you to see how modifications affect a map layout before you recompile the map load module.
You can display the Map Image screen by selecting **Image** option from the **Display** activity on the action bar:

```
Add Modify Compile Delete Display Switch
```

1. Browse
2. Summary
3. Image

If you note any errors while displaying the Map Image screen, you can quickly correct the map while still using MAPC. You can then redisplay the Map Image screen to check your corrections.

When you are satisfied with the modified map layout as displayed on the Map Image screen, you can recompile the map.

Press [PF3] from the Map Image screen to return to the Main Menu.

### Step 5 Recompile the Map

By using MAPC screens, you modified map `XXXMAP` by adding new map fields and changing existing fields. You now need to create an updated load module for the map. You compile the `XXXMAP` map as shown:

#### Compiling the Map

```
Add Modify Compile Delete Display Switch
```

1. Compile
2. View messages

Press [PF3] from the Map Image screen to return to the Main Menu.
To compile the application, position the cursor on the **Compile** item on the action bar and press [Enter]. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing `compile` on the command line and pressing [Enter]

Once you have displayed the **Compile** action item, press [Enter] to compile the map.

When you press [Enter] to recompile the map, MAPC displays messages to indicate whether the map load module recompiled successfully.

- **If the map recompiled successfully**, MAPC a confirming message.
- **If the map could not be recompiled**, MAPC displays an error message. In this case, read the message to determine the problem. After correcting the errors, try again to recompile the map.

**Exit from MAPC**

After you successfully compile your map, you can exit from MAPC by using the **Switch** activity on the Main Menu screen. In this example, you use **Switch** to transfer directly to ADSC to associate modified map XXXMAP with dialogs that use the map:

**Selecting the Switch Activity**

```
Add  Modify  Compile  Delete  Display  Switch
```

```
CA IDMS/DC  Online  Task code  adsct___
-----------------------
CA Int  F3=Exit
_______________________
Map name . . . . . . .  XXXMAP
Map version . . . . .  1
Dictionary name . . . . DEMO
Dictionary node . . . . ______
Screen . . . . . . . . .  1. General options
                                2. Map-Level help text definition
                                3. Associated records
                                4. Layout
                                5. Field definition
```

In this example, ADSCT is the sample task code for ADSC.
Note: You cannot use the switch action unless you entered MAPC using the transfer control facility task code (MAPCT).

Updating Modified Maps In Dialogs Using ADSC

After you modify map XXXMAP, you use ADSC to associate the modified map with dialogs XXXDADD and XXXDUPD. To do this, you will perform the following steps:

1. Retrieve dialog XXXDADD.
2. Recompile the dialog load module.
3. Retrieve and recompile dialog XXXDUPD.

Step 1 Retrieve Dialog XXXDADD

In order to retrieve a dialog load module, you use ADSC.

If you did not transfer directly to ADSC earlier in this section when you exited from MAPC, you need to invoke ADSC by using the task code (for example, ADSCT) for ADSC.

Note: Using the task code ADSCT means that you are invoking ADSC under TCF (the transfer control facility). Once you are in the ADSC tool under TCF, you can switch to another application development tool without returning to DC/UCF.

: P ADSC begins by displaying the Main Menu screen. You use the Main Menu screen to retrieve a dialog definition for update.

Screen Prompts

You typically enter information after one or more of the following Main Menu screen prompts:

- **Dialog name** -- You must specify the name (XXXDADD) that you used when you defined the dialog.
- **Dictionary name** -- You must specify the same dictionary, if any, as you specified for your dialog in Section10, "Defining Dialogs Using ADSC". The correct dictionary name may already be displayed in this field.
- **Dictionary node** -- You must specify the same dictionary node, if any, as you specified for your dialog. The correct dictionary node may already be displayed in this field.
Dialog name . . . . . . . . . xxdadd
Dialog version . . . . . . . 1
Dictionary name . . . . . . . demo
Dictionary node . . . . . . . _______

Screen . . . . . . . . . . . . . 1
1. General options
2. Assign maps
3. Assign database
4. Assign records and tables
5. Assign process modules

Command ==>
Enter F1=Help F3=Exit F10=Action

Press [Enter] to retrieve the dialog.

⚠️ **Note:** If the dialog has not been explicitly released (using the *Release* option of the *Modify* action on the action bar of the Main Menu), naming the dialog on the Main Menu screen retrieves that definition for modification. If the dialog has been released, you use the ADSC Main Menu screen to check out the dialog definition for modification (using the *Checkout* option of the *Modify* action on the action bar of the Main Menu).

If the dialog has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

After you press [Enter], ADSC redisplay the Main Menu screen with either a confirming message or an error message.

Make sure that you typed the correct dialog name, dictionary, and node, as necessary. You can type over any errors and then press [Enter] again.

After you successfully retrieve XXXDADD, you can recompile the dialog load module as described below.

**Step 2 Recompile Dialog XXXDADD**

To include modified map XxxxMAP in dialog XxxxDADD, you need only recompile the dialog:

**Compiling the Dialog**

You compile the XxxxDADD dialog as shown:

```
1. Compile
```
To compile the application, position the cursor on the Compile item on the action bar and press [Enter]. You can position the cursor on Compile by:

- Tabbing to Compile and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to Compile and pressing [Enter]
- Typing compile on the command line and pressing [Enter]

Once you have displayed the Compile action item, press [Enter] to compile the dialog.

When you press [Enter], ADSC attempts to recompile the dialog load module using the new map. ADSC displays a message to indicate whether the dialog has been successfully recompiled or whether errors are present. In the case of errors, read the message to determine the problem. Use ADSC to correct any errors, and then recompile the dialog as described above.

After you successfully recompile dialog XXXDADD, you can retrieve and recompile dialog XXXDUPD as described below.

**Step 3 Retrieve and Recompile Dialog XXXDUPD**

You display dialog XXXDUPD by overtyping the Dialog name on the Main Menu screen and retrieving it.
4. Assign records and tables
5. Assign process modules

Command ===> Enter F1=Help F3=Exit F10=Action

When you have successfully retrieving the dialog, you can immediately recompile the dialog load module.

**Compiling the Dialog**

You compile the XXXDADD dialog as shown:

```
  1. Compile log Compiler
  2. Display messages

```

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

After you successfully recompile dialog XXXDUPD, you can exit from ADSC and return to DC/UCF to execute the application. You press [PF3] from the ADSC Main Menu to exit from ADSC.

**Executing the Application**

At some sites, modified maps are not automatically loaded in the program pool when an old copy of the map already is in the pool.

> **Note:** The new copy of a map load module is loaded automatically in the program pool at your site if the system generation program specifies that NEW COPY IS YES for OLM. The default is NEW COPY IS NO.

This strategy allows users who are executing the old copy of the map to complete their work. When no users are executing the map, developers can make the modified map available for execution. As an application developer, you do this by updating the map in dialogs that use the map and then dynamically loading the modified map in the program pool.
You included modified map XXXMAP in dialogs XXXDADD and XXXDUPD above, in Updating Modified Maps In Dialogs Using ADSC (see page 278). If modified maps are not automatically loaded in the program pool at your site, you now need to dynamically load modified map XXXMAP in the program pool.

Dynamically loading a modified map in the program pool is discussed below, followed by steps for invoking and executing the application.

### Optionally Loading the Modified Map

You can sectionly load modified maps in the program pool by issuing a DCMT VARY PROGRAM NEW COPY command. If you are not sure whether modified maps are automatically loaded at your site, go ahead and issue this DCMT command for your modified map.

#### Warning

**Note:** Do not confuse these commands:

- **DCMT VARY PROGRAM** loads a modified map.
- **DCMT VARY DYNAMIC PROGRAM** dynamically redefines characteristics of programs and maps and can interfere with your ability to execute the map.

You enter the DCMT VARY PROGRAM command in response to the DC/UCF prompt displayed at your site. The following example shows how to enter this command while using CA IDMS/DC:

```
ENTER NEXT TASK CODE:
dcmt vary program demo..xxxmap new copy.
[Enter]
```

If the DCMT command is **successful**, DC/UCF displays a confirming message.

If a different message is displayed, verify that you specified the correct map in the DCMT command. You can type the DCMT command again using the correct map name.

#### Warning

**Note:**

If there are multiple components with the same name, you will be presented with a list of the components and asked to identify the one to be varied.

If you *did* type the correct map name, you can proceed to execute the Department application. In this case, the program pool doesn’t contain a copy of the old map, so a new copy will be loaded automatically the next time a dialog displays the map.
Invoking and Executing the Application

To invoke the Department application from DC/UCF, you enter the task code (XXXDEPT) for the application. For example, when using CA IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:
xxxdept
[Enter]
```

For more information on invoking the Department application, see Instructions For Executing the Application (see page 250).

To test modifications you made to map XXXMAP, you need to execute either dialog XXXDADD or XXXDUPD. According to your application design, each of the following responses selects a function that executes either dialog XXXDADD or XXXDUPD:

- **ADD** selects dialog function ADDDEP. In the final application, ADDDEP will allow end users to add new department information. ADDDEP executes dialog XXXDADD.

- **MOD** selects dialog function MODDEP, which will allow end users to modify existing department information. MODDEP executes dialog XXXDUPD.

- **DEL** selects dialog function DELDEP, which will allow end users to delete existing department information. DELDEP executes dialog XXXDUPD.

### Testing Error Messages

To quickly test out error messages that you defined for map XXXMAP, you can invoke function ADDDEP. The modified map, XXXMAP, is displayed for the function ADDDEP.

Try entering invalid department and department head ID numbers as shown:

```
FUNCTION: ADDDEP
DATE.....: 10/29/99
DEPARTMENT INFORMATION

DEPARTMENT ID .......: aaaa
NAME ......: bbbbbbbbb
HEAD ID ..: cccc

NEXT RESPONSE:
```

When you press [Enter], your error messages are displayed in the map's message field. For example:

```
* ENTER A NUMERIC DEPT ID * * DEPT HEAD IDS ARE NUMERIC *
```

### Verify the Modified Map

You can verify that functions MODDEP and DELDEP also display modified map XXXMAP by displaying these functions. Try transferring from ADDDEP to MODDEP:
Summary 4

You can use MAPC to modify maps during development or at any other time in an application's life cycle. For example, you can modify a map:

- To display information that becomes available as the application is expanded. For example, the name of each department head can be displayed on map XXXMAP when the Department application includes employee data.
- To collect additional information. For example, new government or tax regulations can require the collection of new data.
- To display fields in conformity with end-user requests and site conventions. For example, you modified map XXXMAP in this section for these reasons.

In this section, you used MAPC to modify the layout of map XXXMAP as described below:

1. **You added two new map fields on the Layout screen.** You added the DATE literal field and an adjacent variable field.
2. **You selected fields for editing on the Layout screen.** You used the select-field character (default is %) to select the new variable field and four existing map fields for editing.

3. **You edited the five selected fields on the Field Definition and Literal Definition screens.** You associated the new variable field with AGR-DATE to display the current calendar date in the runtime field.

   You modified the variable field for department ids so that ids display in bright intensity at runtime. You modified the department ID field and the department head id field to define specific error messages for these fields. You then moved the RESPONSE literal field and the adjacent variable field to a different location on the map.

When you recompiled map XXXMAP, MAPC informed you of a critical change on the map. When you make a critical change to a map, you must recompile dialogs that use the map.

In this section, you used ADSC to recompile dialogs XXXDADD and XXXDUPD, which both use map XXXMAP. After you exited from ADSC, you optionally issued a DCMT VARY PROGRAM NEW COPY command for map XXXMAP to load the new map load module in the program pool for execution.

**No Process Statements**

**You then executed the prototype application.** Notice that you do not need to write any process statements to develop a working prototype for an application. This is true regardless of the application's size. An application's user interface can be defined, tested, and tailored for your users before you have any process logic to modify.

End users and application development staff can execute the prototype and implement changes until the prototype is approved. You can then use CA ADS development tools to enhance the application so that it can be used to store, display, modify, and delete data in the application database.

As the application developer, you will enhance the Department application by writing process code and creating work records for dialogs, as described in Part III of this section.

---

### Adding Process Logic to a Dialog

You defined a prototype Department application in Part II of this section. You executed this prototype to test out the flow of control between application functions at runtime. Executing the prototype enabled you to see how screens are displayed to users.

Because you have not yet written any process commands to access the database, the prototype Department application does not retrieve or store data in the database. For example, sample department data that you type on the screen while viewing the ADDDEP function does not get stored in the database when you press [Enter].

In CA ADS, components of a prototype application can be developed directly into the production application. In Part III of this section, you will complete the Department application so that it is fully functional. To do this, you will add modules of process commands to dialogs XXXDADD and XXXDUPD, which you defined in Part II. The process modules that you define for these dialogs will allow users to store, display, modify, and delete department records in the database.

In this section, you will define the two process modules required for dialog XXXDADD. This section includes:
An overview of defining process modules for dialogs

Steps for defining process modules

Steps for adding process modules to dialogs

Steps for executing the application

A summary of what you've accomplished in this section

Because CA ADS is a fourth-generation application development system, major portions of a CA ADS application can be defined without writing any code. For example, you defined the entire Department application prototype in Part II of this section without writing any code.

To enable dialogs to perform runtime processing, you define modules of process commands for the dialogs. For example, you can define process modules to retrieve and display database information, to display messages, to receive input from users, and to evaluate and store valid data.

Process Language

You write process modules by using the CA ADS process language. This language incorporates all the processing capabilities found in a traditional programming language. For example, you can evaluate strings, perform arithmetic functions, and perform conditional tests and loops. Additionally, the CA ADS process language benefits from complete integration with the CA ADS environment.

Categories of Process Commands

The following table lists categories of process commands. For detailed information on process commands, see the CA ADS Reference Section.

<table>
<thead>
<tr>
<th>Category of command</th>
<th>Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic and assignment</td>
<td>Perform calculations and move data</td>
</tr>
<tr>
<td>Conditional</td>
<td>Perform testing and looping</td>
</tr>
<tr>
<td>Control</td>
<td>Specify the next application component executed, govern data passed to that component, and display maps</td>
</tr>
<tr>
<td>Database</td>
<td>Perform database retrieval and update functions and specify recovery options</td>
</tr>
<tr>
<td>Map modification</td>
<td>Request temporary or permanent changes to a map at runtime</td>
</tr>
<tr>
<td>Pageable map</td>
<td>Create, display, and retrieve sets of fields (detail occurrences) for a pageable map</td>
</tr>
<tr>
<td>Queue and scratch management</td>
<td>Define and access temporary disk storage</td>
</tr>
<tr>
<td>Subroutine control</td>
<td>Define and call subroutines</td>
</tr>
<tr>
<td>Utility</td>
<td>Retrieve runtime system status information, request memory dumps, initialize record buffers, direct output to a printer, and display diagnostic information</td>
</tr>
</tbody>
</table>
Process Modules

Process modules can be executed before and after the dialog's map is displayed to the user.

Premap Process

The process module is called a **premap process** when executed before the map. A dialog can have a maximum of one premap process.

A premap process typically includes commands that prepare the map for display. For example, commands in a premap process can retrieve stored values from the database and then display the values along with a message on the dialog's map.

Response Process

The process module is called a **response process** when executed after the map. A dialog can have any number of response processes. The response process executed at runtime is determined by actions that the user takes when inputting data on the map.

A response process typically includes commands that accept end-user input for evaluation and storage.

Declaration Module

A dialog can have a maximum of one declaration module. The module is not executed, but contains declaration statements for SQL that are used during dialog compilation.

For information on declaration statements, see *CA IDMS SQL Programming Section*.

Accessing the Database

Some process modules access the database; for example, to store a new department record.

Process modules accessing a non-SQL defined database can use SQL DML statements or non-SQL DML statements to access that data. Process modules accessing an SQL-defined database can use SQL DML statements.

A dialog using non-SQL DML statements to access a non-SQL defined database must know which portion of the database to access at runtime. You supply this information by adding a predefined subset of the database (that is, a **subschema**) with the dialog. The subschema that you name for dialog XXXDADD, for example, identifies the portion of the sample database that contains the DEPARTMENT record. Subschemas usually are defined by database administrators (DBAs) at a site.

A dialog that uses SQL statements to access an SQL-defined (or non-SQL defined) database will access **tables** that have been defined through SQL statements. To execute the SQL statements, an **access module** must be created based on the SQL statements in one or more programs. The access module is created after dialog compilation; it does not have to be predefined.

**Note:** For more information on programming using SQL DML statements, see *CA IDMS SQL Programming Section* and *CA IDMS SQL Reference Section*. For more information on programming using non-SQL DML statements, see *CA ADS Reference Section* and *CA IDMS Navigational DML Programming Section*. 
Work Records

You also can have work records associated with a map or associated directly with a dialog. A work record does not participate in a subschema. Data defined by work records is not stored in the database. You will not add a work record to dialog XXXDADD.

The following figure shows the components of a fully developed dialog:

- A premap process is executed before the dialog’s map is displayed to the user.
- A response process is executed when the user inputs data on the map.
- A subschema specifies the subset of the application database (schema) that is available to the dialog.
- Work records define data that is used at runtime but not stored in the database.

Integrated Data Dictionary

As an application developer, you define premap and response processes by using the Integrated Data Dictionary (IDD) menu facility.

You will use IDD in this section to define one premap process (XXXDADD-PREMAP) and one response process (XXXDADD-RESPONSE) for dialog XXXDADD:

Dialog Compiler
When you finish using IDD to define process modules, you use the **CA ADS Catalog compiler (ADSC)** to associate the process modules with dialogs. For example, after you use IDD to define process modules `XXXDADD-PREMAP` and `XXXDADD-RESPONSE`, you use ADSC to associate these process modules with dialog `XXXDADD`.

### Enhancing Sample Application Dialogs

In this section, you will enhance dialog `XXXDADD` by adding a premap process, response process, and subschema to the dialog, as shown below. Process modules `XXXDADD-PREMAP` and `XXXDADD-RESPONSE` perform all processing required for dialog `XXXDADD`. Subschema `EMPSS01` specifies the portion of the database available to process `XXXDADD-RESPONSE`, which stores department data in the database.

<table>
<thead>
<tr>
<th>Process name</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>XXXDADD-PREMAP</code></td>
<td>Premap</td>
<td>Displays the dialog's map with a message</td>
</tr>
<tr>
<td><code>XXXDADD-RESPONSE</code></td>
<td>Response</td>
<td>Accepts department information supplied by the end user on the map; stores new department information in the database; redisplays the stored input to the user with a confirming message</td>
</tr>
</tbody>
</table>

Steps for defining process modules and associating the modules and a subschema with a dialog are discussed below.
Defining Process Modules Using IDD

Contents

- Step 1 Invoke the IDD Menu Facility (see page 291)
- Step 2 Define Process Module XXXDADD-PREMAP (see page 292)
- Step 3 Define Process Module XXXDADD-RESPONSE (see page 296)
- Step 4 Exit from IDD (see page 299)

You define process modules in the data dictionary. You can associate a given process module with any number of dialogs. You use the **IDD menu facility** to define process modules in the data dictionary. A typical IDD menu facility screen is shown below.

```
CA CAGJF0
- IDD REL 15.0 *** MASTER SELECTION *** TOP

DICTIONARY NAME...: DEMO NODE NAME...
USER NAME.........: PASSWORD........:
USAGE MODE.........: X UPDATE RETRIEVAL
PFKEY SIMULATION..: X OFF ON

ATTR = ATTRIBUTE <PF2> PROC = PROCESS <PF3>
CLAS = CLASS <PF4> PROG = PROGRAM <PF5>
ELEM = ELEMENT <PF6> RECD = RECORD <PF7>
FILE = FILE <PF8> TABL = TABLE <PF9>
MODU = MODULE <PF10> USER = USER <PF11>
ENTL = USER DEFINED ENTITY LIST USER = USER
MSGS = MESSAGE SYST = SYSTEM
QFIL = QFILE OPTI = OPTIONS
DISP = DISPLAY ALL HELP = HELP
```

- **Heading and system message area** -- Names the IDD screen and returns messages to you.
- **Command area** -- Allows you to enter commands and IDD screen names.
- **Specification area** -- Prompts you for specifications.
- **Activity selection area** -- Allows you to select the next IDD activity from a list of available activities.

If you need any more information on IDD while using the IDD menu facility, see .

**Steps**

To define process modules XXXDADD-PREMAP and XXXDADD-RESPONSE, you will perform the following steps:

1. Invoke the IDD menu facility.
2. Define process module XXXDADD-PREMAP.
3. Define process module XXXDADD-RESPONSE.

4. Exit from the IDD menu facility.

These steps are described below. Steps for adding these modules to dialog XXXDADD are presented later in this section.

Step 1 Invoke the IDD Menu Facility

You can invoke IDD from DC/UCF by entering the task code for the IDD menu facility (for example, IDDMT) in response to the prompt presented by DC/UCF. For example, you can invoke IDD from CA IDMS/DC as shown:

```
ENTER NEXT TASK CODE: iddmt [Enter]
```

For more information on task codes for CA ADS development tools, see Application Development Tools (see page 180).

Master Selection Screen

IDD begins by displaying the Master Selection screen:

```
CA
CAGJF0
IDD REL 15.0 *** MASTER SELECTION *** TOP
->
```

```
DICTIONARY NAME...: DEMO
NODE NAME...:
USER NAME............:
PASSWORD............:
USAGE MODE..........: X UPDATE _ RETRIEVAL
PFKEY SIMULATION...: X OFF _ ON
  _ ATTR = ATTRIBUTE <PF2>  _ PROC = PROCESS <PF3>
  _ CLAS = CLASS <PF4>  _ PROG = PROGRAM <PF5>
  _ ELEM = ELEMENT <PF6>  _ REC = RECORD <PF7>
  _ FILE = FILE <PF8>  _ TABLE = TABLE <PF9>
  _ MODU = MODULE <PF10>  _ USER = USER <PF11>
  _ ENTL = USER DEFINED ENTITY LIST _ SYST = SYSTEM
  _ MSGS = MESSAGE _ OPTI = OPTIONS
  _ OFIL = OFILE _ HELP = HELP <PF1>
  _ DISP = DISPLAY ALL
```

Screen Prompts

When you start an IDD menu facility session, you may need to sign on to IDD. To do so, supply information after one or more of the following Master Selection screen prompts, as appropriate:

- **DICTIONARY NAME** -- You must specify the same dictionary, if any, as you specified for all other Department application components. The correct dictionary name may already be displayed in this field.

- **NODE NAME** -- You must specify the same dictionary node, if any, as you specified for all other Department application components. The correct dictionary node may already be displayed in this field.
• **USER NAME** -- You may need to supply your user ID after this prompt. You can check with others at your site to see if you are required to sign on to the IDD menu facility. If you have a user ID, you can go ahead and enter it, just in case.

• **PASSWORD** -- If you need to supply signon information for IDD, you also may need to enter your password after this prompt. Your password is not displayed on the screen when you type it. If you have a password, you can go ahead and enter it.

**Signing On**

You can sign on to the IDD menu facility as shown:

```
CA IDMS - 19.0

IDD REL 15.0

CA

*** MASTER SELECTION ***

->

DICTIONARY NAME....: DEMO

USER NAME..........: xxxx

PASSWORD..........:

USAGE MODE........: X UPDATE _ RETRIEVAL

PFKEY SIMULATION..: X OFF _ ON

When you are successfully signed on to the IDD menu facility, the following message is displayed on the Master Selection screen:

SIGNON TO IDD WAS SUCCESSFUL

From the Master Selection screen, you can display the Process Entity screen and define process module XXXDADD-PREMAP, as described in Step 2.

**Step 2 Define Process Module XXXDADD-PREMAP**

In this step, you will define process module XXXDADD-PREMAP. This process module displays the dialog's map with a message. This processing is performed by the DISPLAY command, as shown below.

```
DISPLAY MSG TEXT

'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
```

The map is displayed the message that is defined here between single quotation marks. In this example, the DISPLAY command is entered on two lines and is ended by a period (.).

You use the following IDD menu facility screens to define process modules:

1. Use the **Process Entity screen** to specify basic information on the process module, including the module's name.

2. Use the **Process Source screen** to enter source commands for the process module.

**Process Entity Screen**

To begin defining process module XXXDADD-PREMAP, you display the Process Entity screen in any of the following ways:
Type the identifier for the screen in the command area and press [Enter].

Type a nonblank character in front of a screen identifier and press [Enter].

Press a control key to display the associated screen.

```
   CA CAGJF0
  --> proc
```

SIGNON TO IDD WAS SUCCESSFUL

DICTIONARY NAME...: DEMO
NODE NAME...:

USER NAME........:
PASSWORD........:

USAGE MODE.......: X UPDATE _ RETRIEVAL

PFKEY SIMULATION...: X OFF _ ON

- ATTR = ATTRIBUTE <PF2> x PROC = PROCESS <PF3>
- CLAS = CLASS <PF4> - PROG = PROGRAM <PF5>
- ELEM = ELEMENT <PF6> - RECD = RECORD <PF7>
- FILE = FILE <PF8> - TABL = TABLE <PF9>
- MODU = MODULE <PF10> - USER = USER <PF11>
- ENTL = USER DEFINED ENTITY LIST - SYST = SYSTEM
- MSGS = MESSAGE
- QFIL = QFILE
- DISP = DISPLAY ALL _ HELP = HELP <PF1>

Screen Prompts

When you define a process module, you usually specify information for the following Process Entity screen prompts:
- **PROCESS NAME** -- You must supply a process module name. The name that you specify must be unique.

- **DISPLAY** -- You deselect the DISPLAY action when you intend to add a new process module. To do this, type a blank over the X displayed to the left of the action.

- **ADD** -- You select the ADD action to specify that you are defining a new process module. To do this, type a nonblank character to the left of the action.

- **DESCRIPTION** -- You optionally type a brief description of the process module.

**Defining the XXXDADD-PREMAP Process Module**

You define the above basic information for a process module by using the Process Entity screen. For example, you define process module XXXDADD-PREMAP by:

- Deselecting the DISPLAY action by typing a space over the X to its left.
- Typing the name for the process module. (You can use your initials instead of XXX.)
- Selecting the ADD action.
- Optionally typing a description of the process.

```
IDD REL 15.0
*** PROCESS ENTITY ***
PROC
-> DICT=DEMO

DISPLAY
MODIFY
PROCESS NAME....: xxdadd-premap

ADD
VERSION NUMBER..: 1

DELETE

DESCRIPTION.....: display map to add departments
```

When you press [Enter], IDD redisplays the Process Entity screen with a message:

- **If the process module is successfully defined**, the Process Entity screen displays a message like:

```
PROCESS 'XXXDADD-PREMAP' VERSION 1 ADDED
```

- **If the process module cannot be defined**, the Process Entity screen displays a different message than the message indicated above.

Read the message to determine the problem. You can type over any errors, and then press [Enter] again.
Entering Source Statements

After you specify basic information on a process module, you use the Process Source screen to enter process commands for the process module.

To display the Process Source screen:

- Type the identifier for the screen in the command area and press [Enter].
- Type a nonblank character in front of a screen identifier and press [Enter].
- Press a control key to display the associated screen.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>PROCESS NAME....: XXXDADD-PREMAP</td>
</tr>
<tr>
<td>modify</td>
<td>VERSION NUMBER..: 1 HIGHEST NEXT HIGHEST</td>
</tr>
<tr>
<td>delete</td>
<td>LOWEST NEXT LOWEST</td>
</tr>
<tr>
<td>description</td>
<td>DESCRIPTION.....: DISPLAY MAP TO ADD DEPARTMENTS</td>
</tr>
</tbody>
</table>

Entering Process Statements

Enter the process statements in the text entry area.

Important! Do not type any characters beyond column 72.

Considerations

- Enclose the message text in single quotation marks.
- End the DISPLAY command with a period.
After pressing [Enter], the Process Source screen is redisplayed. Look over the redisplayed process statement for possible syntax errors in the process code.

**Note:** IDD does not compile the process code; the code is compiled when the process module is associated with a dialog under ADSC.

```
IDD REL 15.0            *** PROCESS SOURCE ***             SRCE
->                    PAGE 1 LINE 1                     1/2
PROCESS 'XXXDADD-PREMAP' VERSION 1 MODIFIED

DISPLAY MSG TEXT
'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
```

After you press [Enter], IDD adds the process module to the data dictionary and redispalyes the Process Source screen with a message like:

PROCESS 'XXXDADD-PREMAP' VERSION 1 MODIFIED

**Syntax Errors**

It is a good idea to look over the redisplayed commands for syntax errors. This is because your syntax does not get compiled until you add the process module to a dialog by using ADSC. Typical syntax errors include:

- Process statements that extend beyond column 72
- Omitted keywords
- Misspelled comments or record element names
- Omitted periods
- Omitted or misplaced single quotation marks
- Single quotation marks entered as double quotation marks
- Double quotation marks entered as single quotation marks within a quoted string

If you make any errors on the Process Source screen, you can type over them and press [Enter] again.

After you finish using the Process Source screen for process module XXXDADD-PREMAP, you can proceed to define process module XXXDADD-RESPONSE.

**Step 3 Define Process Module XXXDADD-RESPONSE**

Process module XXXDADD-RESPONSE processes data entered by the user. To add a department record to the database, source commands for process module XXXDADD-RESPONSE need to:

1. **Verify that the department is new.** To do this, the process module must determine that the unique department ID specified by the user does not already exist in the database.
2. **Add the department record to the database** if it is a new record.

3. **Redisplay the screen with a confirming message** and allow the user to add another department record.

Additionally, the process module needs to be able to transfer the user to another dialog function (for example, to MODDEP) when requested by the user.

**Commands for XXXDADD-RESPONSE**

Below are the sample commands for process module XXXDADD-RESPONSE. These commands handle all of the above processing. Notice that CA ADS does not require you to code MOVE statements for data in this case because the record displayed on the screen is the database record itself. Also notice that one STORE command stores the entire Department record in the database. You define a message for display to the user by including the message in the DISPLAY command. In this example, the DISPLAY command is entered on two lines and is ended by a period (.). This process module evaluates data input by the user and stores new department records in the database.

```
READY USAGE-MODE UPDATE.
  IF AGR-CURRENT-RESPONSE NE SPACES
    AND FIELD DEPT-ID-0410 NOT CHANGED
    THEN
      EXECUTE NEXT FUNCTION.
  OBTAIN CALC DEPARTMENT.
    IF DB-REC-NOT-FOUND
      THEN DO.
      STORE DEPARTMENT.
      DISPLAY MSG TEXT 'DEPARTMENT ADDED'.
      END.
    DISPLAY MSG TEXT 'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

1 If the user enters a valid response name (such as MOD) in the map’s $RESPONSE field and doesn’t try to add a new department, the runtime system terminates this response process and transfers execution to the function (such as MODDEP) requested by the user. (The runtime system stores valid responses in system field AGR-CURRENT-RESPONSE.)

2 This statement attempts to locate the specified department in the database.

3 If the specified department is not in the database, these statements store the department in the database and then redisplay the screen with the DEPARTMENT ADDED message.

4 If the specified department already exists in the database, this statement redispaly the screen with the TRY AGAIN error message.

**Specifying Basic Information for the Process Module**

You start defining process module XXXDADD-RESPONSE by using the IDD Process Entity screen to define basic information for the process module.
Display a blank Process Entity screen.

```
IDD REL 15.0 *** PROCESS ENTITY ***
-> proc
PROC DICT=DEMO
DISPLAY PROCESS NAME.....: xxdadd-response
MODIFY ADD VERSION NUMBER..: 1     HIGHEST    NEXT HIGHEST
     DELETE            _ LOWEST     _ NEXT LOWEST
DESCRIPTION.....: test input and add new department
```

After you press [Enter], the Process Entity screen is redisplayed with a message like:

```
PROCESS 'XXXDADD-RESPONSE' VERSION 1 ADDED
```

This message indicates that your process module specifications have been added to the data dictionary.

To continue defining process module XXXDADD-RESPONSE, you use the Process Source screen to enter source commands.

**Display the Process Source Screen**

```
IDD REL 15.0 *** PROCESS SOURCE ***
-> srce
PROC PROCESS 'XXXDADD-RESPONSE' VERSION 1 ADDED
DISPLAY PROCESS NAME.....: XXXDADD-RESPONSE
MODIFY ADD VERSION NUMBER..: 1     HIGHEST    NEXT HIGHEST
     DELETE            _ LOWEST     _ NEXT LOWEST
DESCRIPTION.....: TEST INPUT AND ADD NEW DEPARTMENT
```

Enter process statements on the Process Source screen.

**Considerations**

- Periods and single quotation marks shown on this screen are required.
- Don't extend statements beyond column 72.
- You can enter blank lines in your source to improve readability.
- You can type spaces to indent lines, making source statements easier to read and debug.
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
   EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
   THEN DO.
      STORE DEPARTMENT.
      DISPLAY MSG TEXT
         'DEPARTMENT ADDED'.
   END.

DISPLAY MSG TEXT
   'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.

After you press [Enter], the Process Source screen is redisplayed. Look over the redisplayed process statements for possible syntax errors. Make sure that the period is outside the single quote on the message.

After you press [Enter], IDD adds the process module to the data dictionary and redisplays the Process Source screen with a message like:

PROCESS 'XXXDADD-RESPONSE' VERSION 1 MODIFIED

This message indicates that the entry for your process module in the data dictionary has successfully been modified.

If you notice any errors on the redisplayed screen, you can type over the errors to correct them, and then press [Enter] again.

After you finish defining process module XXXDADD-RESPONSE, you can exit from IDD.

Step 4 Exit from IDD

When you are using the IDD menu facility under the transfer control facility (TCF), you can exit from IDD by using the SWITCH command.
You can use SWITCH to either transfer to another development tool or return to DC/UCF. In the this sample session, you will transfer to ADSC so that you can associate process modules XXXDADD-PREMAP and XXXDADD-RESPONSE with dialog XXXDADD.

To transfer to ADSC, enter the task code for ADSC (for example, ADSCT) along with the SWITCH command in the command area of any IDD menu facility screen:

```
IDD REL 15.0 *** PROCESS SOURCE *** PROCESS 'XXXDADD-RESPONSE' VERSION 1 MODIFIED
SRCE -> adsct PAGE 1 LINE 1
```

```
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
EXECUTE NEXT FUNCTION.
OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.
STORE DEPARTMENT.
DISPLAY MSG TEXT
'DEPARTMENT ADDED'.
END.
DISPLAY MSG TEXT
'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

### Adding Process Modules to Dialogs Using ADSC

You use ADSC to add process modules XXXDADD-PREMAP and XXXDADD-RESPONSE to dialog XXXDADD. You associate a subschema with the dialog to make a subset of the database available to dialog XXXDADD and its process modules. Additionally, you specify dialog options to help you debug the dialog.

- Step 1 Retrieve Dialog XXXDADD (see page 300)
- Step 2 Specify Dialog Options (see page 302)
- Step 3 Add a Subschema (see page 303)
- Step 4 Add Process Modules (see page 307)
- Step 5 Recompile the Dialog (see page 310)
- Correct Errors in Process Modules (see page 311)
- Display structural messages (see page 312)
- Display diagnostic messages (see page 312)
- Correct structural errors (see page 315)
- Correct syntax errors (see page 316)
- Update dialogs that use the process module (see page 317)

Steps for viewing and correcting compile-time errors in process modules are discussed later in this section, see Correct Errors in Process Modules (see page 311).

### Step 1 Retrieve Dialog XXXDADD

In order to retrieve a dialog, you use ADSC.
If you did not transfer to ADSC earlier in this section when you exited from IDD, you need to invoke ADSC by using the task code (for example, ADSCT) for ADSC.

ADSC begins by displaying the Main Menu screen. You use a blank Main Menu screen to retrieve a dialog.

Screen Prompts

You typically enter information after one or more of the following Main Menu screen prompts:

- **Dialog name** -- You must specify the name (XXXDADD) that you used when you defined the dialog.

- **Dictionary name** -- You must specify the same dictionary, if any, as you specified for your dialog in Section 4. The correct dictionary name may already be displayed in this field.

- **Dictionary node** -- You must specify the same dictionary node, if any, as you specified for your dialog definition. The correct dictionary node may already be displayed in this field.

You use the ADSC Main Menu screen to retrieve dialog XXXDADD:

```
Add Modify Compile Delete Display Switch

CA ADS Online Dialog Compiler
CA, Inc.
```

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

Copyright (C) 2003 CA, Inc.
Command ===> Enter F1=Help F3=Exit F10=Action
```

Press [Enter] to retrieve the application.

After you press [Enter], ADSC redispplays the Main Menu screen with a message confirming that the dialog is available for modification.

**Note:** If the dialog has not been explicitly released (using the Release option of the Modify action on the action bar of the Main Menu), naming the dialog on the Main Menu screen retrieves that definition for modification. If the dialog has been released, you use the ADSC Main Menu screen to check out the dialog definition for modification (using the Checkout option of the Modify action on the action bar of the Main Menu).
For information on checking out a dialog, see *CA ADS Reference Section*.

If the dialog has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

After you press [Enter], ADSC displays dialog XXXDADD on the Main Menu screen, with appropriate messages.

You can now specify dialog options on the Options and Directives screen. You access the Options and Directives screen by choosing option 1 from the Main Menu.

**Step 2 Specify Dialog Options**

You use the **Options and Directives** screen to specify options to help you develop and debug dialogs. A sample Options and Directives screen is shown below:

**Sample Options and Directives Screen**

<table>
<thead>
<tr>
<th>Options and Directives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog XXXDADD Version 1</td>
</tr>
</tbody>
</table>

- **Message prefix**... DC
- **Autostatus record**... ADSO-STAT-DEF-REC
- **Version**... 1
- **Options and directives**... _Mainline dialog_
  _Symbol table is enabled_
  _Diagnostic table is enabled_
  _Entry point is premap_
  _COBOL moves are enabled_
  _Activity logging_
  _Retrieval locks are kept_
  _Autostatus is enabled_

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

**Screen Prompts**

During development, you typically use the following Options and Directives screen prompts to enable options for the dialog:

- **Symbol table is enabled** -- You can create a symbol table to detect, trace, and resolve programming errors in dialogs.
  The symbol table stores information on the dialog, such as record element names and internal line numbers for commands in process modules. This information allows the **online debugger** to track, set breakpoints in, and alter execution of a dialog.
  For more information on using the online debugger with CA ADS, see *CA ADS Reference Section*. 
Diagnostic table is enabled -- You can create a diagnostic table to help locate the causes for dialog abends during development. When a dialog abends, the dialog's diagnostic table allows the Dialog Abort Information screen to display the process command that was being executed when the abend occurred. For more information on diagnostic tables, see CA ADS Reference Section.

When the application is ready for production use, you can disable dialog symbol and diagnostic tables and disable display of the Dialog Abort Information screen.

Specifying XXXDADD Options

To specify development options for dialog XXXDADD, display and use the Options and Directives screen. Enable the symbol table and diagnostic table.

Options and Directives

Dialog XXXDADD Version 1

Message prefix . . . . . . . . . . . . DC
Autostatus record . . . . . . . . . . . . ADSO-STAT-DEF-REC
Version . . . . . . . . . . . . . . . . . 1
Options and directives . . . . . Mainline dialog
/ Symbol table is enabled
/ Diagnostic table is enabled
/ Entry point is premap
/ COBOL moves are enabled
/ Activity logging
/ Retrieval locks are kept
/ Autostatus is enabled

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

After you press [Enter], ADSC displays a confirming message on the Options and Directives screen to inform you that your specifications contain no errors.

You are now ready to add database information to the dialog definition. Return to the Main Menu by pressing [PF3].

Step 3 Add a Subschema

Using Non-SQL DML Statements to Access a Database

A non-SQL defined database is defined by a schema. A schema typically includes definitions for the database records required by your application. For example, the schema for a fully developed personnel application probably would include records for employee information, job descriptions, and hospital and dental insurance information.

To promote efficient use of the database and other system resources at runtime, you restrict each dialog to a specific subset of the database. Each subset of the database is defined by a subschema. The subschema identifies the subset of the database that the dialog can access at runtime.
The way that a subschema relates a dialog to the application’s database is shown below. Schemas and subschemas usually are defined by database administrators (DBAs) or system administrators, based on application requirements. A subschema determines the portion of the database (schema) available to the dialog at runtime.

**IDMSDB--Step 3: Add a Subschema**

You need to associate a subschema with each dialog that accesses a non-SQL defined database using non-SQL DML statements. For example, you will associate a subschema with dialog XXXDADD because commands in process module XXXDADD-RESPONSE access the database to store new department information using non-SQL DML statements.

You associate the subschema with the dialog before you add process XXXDADD-RESPONSE so that ADSC can verify the process module’s database commands.

You use the **Database Specifications** screen to associate a subschema with dialog XXXDADD.

**Using SQL Statements to Access a Database**

An SQL-defined application database is defined by tables associated with a schema. A schema typically includes definitions for the tables required by your application. For example, the schema for a fully developed personnel application probably would include tables for employee information, job descriptions, and hospital and dental insurance information.

To promote efficient use of the database and other system resources at runtime in the SQL environment, you identify an **access module** to be associated with the dialog. An access module identifies the method of access that the dialog will use at runtime.

Access modules are made up of relational command modules (RCMs), and are usually created by the application developer.

For information on creating an access module, see *CA IDMS SQL Programming Section*.

**Modifying the Sample Dialogs**
In the XXXDADD and XXXDUPD dialogs, you are accessing a non-SQL defined database. Therefore, you must associate a subschema with each of these dialogs.

You use the **Database Specifications** screen to associate a subschema with dialog XXXDADD.

**Accessing the Database Specifications Screen**

To access the Database Specifications screen, you enter **3** at the **Screen** prompt on the Main Menu.

```
Dialog name . . . . . . XXXDADD
Dialog version . . . . 1
Dictionary name . . . . DEMO
Dictionary node . . . . ______
Screen . . . . . . . . . 3
```

The Database Specifications screen is displayed.

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

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

**Screen Prompts**
To associate a subschema with a dialog, you use the following Database Specifications screen prompts:

- **Subschema** -- You name an existing subschema in response to this prompt. A sample subschema name (EMPSS01) is used in this section; a different name may be required at your site.

- **Schema** -- You may need to name the schema for your application in response to this prompt. Naming a schema usually is required only when your schema is not unique. For example, schemas typically are not unique when duplicate, identical development and production databases are defined. If you know the name of your schema, go ahead and specify it here.

**Modifying XXXDADD Dialog**

Modify the XXXDADD dialog to include the subschema EMPSS01.

```
Database Specifications
Dialog XXXDADD Version 1

Subschema ............. empss01
Schema .................
Version ...............

Access Module ............. XXXDADD
SQL Compliance ........... _ ANSI-standard SQL

Date Default Format ....... 1. ISO
Time Default Format ....... 2. USA
3. EUR
4. JIS
```

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

**Note:** By default, the access module is given the same name as the dialog.

When you press [Enter], ADSC associates the named subschema with the dialog if there are no errors, and then redisplays the screen with a confirming message.

A different message is displayed if ADSC detects any errors.

Read the message to determine the problem. Make sure that you have specified the correct subschema on the Database Specifications screen. If you didn't specify a schema name, ask others at your site whether a schema name is required. After you change information on the screen, press [Enter] again.

If the error persists, verify that you specified the correct version of the subschema.
You are now ready to add process modules to the dialog definition. Return to the Main Menu by pressing [PF3].

**Step 4 Add Process Modules**

**Premap Processes**

You add a **premap process** to a dialog so the dialog can perform processing or access database information before the dialog's map is displayed. If a dialog has a premap process, that process is executed as soon as the dialog begins.

You add a premap process to a dialog by using the ADSC **Process Modules** screen.

**Response Processes**

A **response process** enables a dialog to perform processing after the user inputs data on the map.

You can add any number of response processes to a dialog, enabling a dialog to perform many different processing operations. At runtime, the response process executed for the dialog is determined by actions taken by the user during dialog execution, as described later in this step.

You add a response process to a dialog by using the ADSC **Process Modules** screen.

**Accessing the Process Modules Screen**

You access the Process Modules screen by entering 5 at the **Screen** prompt on the Main Menu and pressing [Enter].

**Sample Process Modules Screen**

The Process Modules screen is displayed.

<table>
<thead>
<tr>
<th>Process Modules</th>
<th>Page 1 of 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog XXXDADD</td>
<td>Version 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Key</th>
<th>Value</th>
<th>Type</th>
<th>Execute on errors</th>
<th>Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

**Screen Prompts**

You specify information on the process modules after prompts on the Process Modules screen:
- **Name** -- You specify a process module name after the Name prompt. By doing this, you associate the process module with the dialog.

- **Version** -- You specify the version number of the associated process module as specified in the dictionary.

- **Type** -- You specify the type of process module you have just named.
  - 1 indicates that this is a declaration module used with SQL only. 
    
    >> For information on declaration modules, see *CA IDMS SQL Programming Section*.
  
  - 2 indicates that this is a premap process.
  
  - 3 indicates that this is a response process.
  
  - 4 indicates that this is the default response process for the dialog.
    
    At runtime, the dialog's default response process (if any) is executed when the user inputs information on the dialog's map without specifying any response process to be executed. 
    
    You will define a premap process and a response process for dialog XXXDADD. You will not define a default response process.

- **Key** -- If the process module is defined as a response process, you can associate a key (for example, [Enter] or [PF1]) with the process.
  
  At runtime, the response process is executed for the dialog if the user presses the associated control key while viewing the dialog's map.
  
  You will associate [Enter] with response process XXXDADD-RESPONSE.

- **Value** -- If the process module is defined as a response process, you can associate a value (for example, MOD) with the process.
  
  At runtime, the response process is executed when the dialog's user enters the response field value in the screen's $RESPONSE field.
  
  For example, assume that you defined response process XXXDADD-MOD to be executed when the user requests transfer to function MODDEP. In this case, you would give response process XXXDADD-MOD a response field value of MOD. This way, MOD first executes XXXDADD-MOD, which then transfers control to function MODDEP.
  
  You will not add a response field value to response process XXXDADD-RESPONSE.

**Using ENTER as a Key**

When you add response process XXXDADD-RESPONSE to dialog XXXDADD, you will associate the process with [Enter]. ENTER is a good key for this response process because users are accustomed to pressing [Enter] to input information, and are more likely to press [Enter] than a PF key when unfamiliar with the application.

At runtime, whenever the user presses [Enter] to input data for dialog XXXDADD, response process XXXDADD-RESPONSE is executed.

ENTER is often associated with the response process that performs the dialog's major processing. To make it easier for users to input information, the runtime system *automatically executes* a dialog's ENTER response process when the user inputs information without otherwise specifying a response process to execute.
In the sample application, when using dialog XXXDADD, response process XXXDADD-RESPONSE is executed if the user:

- Inputs a new department record by pressing [Enter]
- Requests transfer to the MODDEP function by using the MOD response or by pressing [PF2]. This is true in this dialog because:
  
  1. Response process XXXDADD-RESPONSE is associated with [Enter].
  2. No response process is associated with MOD or with [PF5], both of which are valid the current function (ADDDEP).

**Immediately Executable Function**

As an application developer, you can inhibit execution of XXXDADD-RESPONSE when the user requests transfer to MODDEP. To do this, you make function MODDEP an immediately executable function.

At runtime, when the user requests transfer to an immediately executable function, control transfers immediately to that function. No response process is executed before transfer occurs.

For more information on immediately executable functions, see the *CA ADS Reference Section*.

**The EXECUTE NEXT FUNCTION Command**

In this section, you will not make function MODDEP an immediately executable function. Instead, you will enable response process XXXDADD-RESPONSE to transfer control. To do this, you include an EXECUTE NEXT FUNCTION command in the process module. When executed, this command transfers control to the function specified by the user.

For example, when you defined XXXDADD-RESPONSE by using IDD earlier in this section, you included the EXECUTE NEXT FUNCTION command in the following conditional structure:

```
IF AGR-CURRENT-RESPONSE NE SPACES
AND DEPT-ID-0415 NOT CHANGED
THEN
  EXECUTE NEXT FUNCTION. -- Control transfers to the next function only when the above two conditions are met.
```

AGR-CURRENT-RESPONSE is an element in the system-supplied ADSO-APPLICATION-GLOBAL-RECORD that the runtime system uses for flow-of-control processing.

This conditional structure causes the following different events to occur at runtime:

- **A department is added and control remains in dialog XXXDADD** whenever the user enters new department information.
- **Control transfers** when the user enters a valid response name (AGR-CURRENT-RESPONSE NE SPACES) without entering a new department ID.

**Adding Process Modules to the Dialog**

You use the Process Modules screen to add two processes to the dialog:
Step 5 Recompile the Dialog

In this section, you have enhanced dialog XXXDADD by adding a subschema, premap process, and response process to the dialog definition. To update these modifications to the dialog load module, you must recompile the process modules and recompile the dialog by selecting the compile activity from the action bar on the Main Menu:

```
Add  Modify  Compile  Delete  Display  Switch
```

```
1 1. Compile  log Compiler
2. Display messages
------------------------- international, Inc.
F3=Exit
```

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

Command ==> 

Enter F1=Help F3=Exit F10=Action
After you press [Enter] to recompile the dialog, ADSC compiles the process module source code. ADSC displays a confirming message if no errors are found. This indicates that the compiled process modules were successfully added to the dialog. If there are no errors, ADSC then creates the dialog load module and redisplays the Main Menu.

**Error Messages**

If ADSC finds errors while compiling a process, it displays an error message. In this case, you display and correct errors in the process module as discussed in *Correct Errors in Process Modules* (see page 311), later in this section.

Different messages are displayed depending on the nature of the error. Read the message to determine the problem. Verify that you have correctly typed the process module name on the Process Modules screen. You can type over errors on the screen, and then press [Enter] again.

After you successfully recompile dialog XXXDADD, you can exit from ADSC by pressing [PF3].

⚠️ **Note:** When a dialog is added or checked out, a queue is established. The queue is deleted only when:

- The dialog is released with no uncompiled changes
- The dialog is deleted

A dialog with changes that is released can be retrieved by another developer, but the queue remains.

**Correct Errors in Process Modules**

When you add a premap or response process to a dialog, ADSC compiles the related process module commands. A fully compiled copy of the process module is stored in the dialog.

**Compile Time Errors**

When errors arise at compile time:

1. ADSC inserts diagnostic messages in the dialog's copy of the process module.
2. ADSC redisplays the Main Menu screen, as appropriate, with a message indicating that there is an error in a particular process module.

In this case, the process module is *not added* to the dialog. If ADSC indicates that there are compile errors, you must:

1. **Display diagnostic messages** for the process module.
2. **Correct errors**, including:

- **Discrepancies** between the process module and other dialog components
- **Syntax errors** in the process module source.

3. **Recompile the process module** and update the process module in the dialog by recompiling the dialog load module.

### Display structural messages

Some errors occur from the definition of the process module in ADSC. These are called **structural errors**.

When a message indicates that there are structural errors in your definition, access the **Structural Error Display** screen using the **Display messages** option from the **Compile** activity on the action bar.

![Dialog Process Source Screen](image)

Select 2, then Enter to see structural errors.

**Command ==>**

Enter F1=Help F3=Exit F10=Action

**Structural Error Display Screen**

DC498163 Response process XXXDADD-PREMAP must have a PFKEY or $Response

Use this screen to find discrepancies between the process module and other components of the dialog. A missing or incorrect dialog component (for example, a subschema, map, or work record) causes compile-time errors for process commands that reference the component.

### Display diagnostic messages

You display the ADSC **Dialog Process Source** screen to view a listing of the process module with diagnostic messages. You use this screen to find syntax errors in process module commands. Syntax errors (for example, mistyped commands and omitted periods) are the most frequent cause of compile-time errors.

**Accessing the Dialog Process Source Screen**
To access the Dialog Process Source screen,

When a message indicates that there are compile errors in your process code, use the **Display messages** option from the **Compile** activity on the action bar first to access the Compiled Process Modules screen.

```
Add  Modify  Compile  Delete  Display  Switch
```

Add Modify Compile Delete Display Switch

<table>
<thead>
<tr>
<th>Number</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compile</td>
</tr>
<tr>
<td>2</td>
<td>Display messages</td>
</tr>
</tbody>
</table>

**Enterprise, Inc.**

F3=Exit

Dialog name . . . . . . XXXDADD
Dialog version . . . . . . 1
Dictionary name . . . . . . DEMO
Dictionary node . . . . . . __________

Screen . . . . . . . . . . . . 1
1. General options
2. Assign maps
3. Assign database
4. Assign records and tables
5. Assign process modules

Select 2, then Enter to see compile errors.

Command ==>

Enter F1=Help F3=Exit F10=Action

**Compiled Process Modules Screen**

Compiled Process Modules Page 1 of 1

Dialog XXXDADD Ver 1

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Type</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXDADD-PREMAP</td>
<td>0001</td>
<td>3</td>
<td>PF3</td>
<td>Value</td>
</tr>
</tbody>
</table>

Name __________________________
Version ____ Type ____________
Key _____ Value ____________

Type: 1=Declaration 2=Premap 3=Response 4=Default Response
Select a process for Display or Print.

F1=Help F3=Exit F7=Bkwd F8=Fwd F11=Dialog-level messages

The Compiled Process Modules screen shows the total number of commands in the process module and the number of errors encountered. You have the option of displaying the process code with the errors noted or printing them.

To display the process code, select 2 next to the process module you want to see and press [Enter].

The Dialog Process Source screen is then displayed.
Dialog Process Source Screen

1. _____________________________________________________________________________.

<PROCESS> XXXDADD-PREMAP 0001

100 DISPLAY MSG TEXT
   'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
   $

< E > DC157001 INVALID INITIATING KEYWORD FOR COMMAND. STMT FLUSHED.

2. _____________________________________________________________________________.

F3=Exit  F5=IDD  F7=Bkwd  F8=Fwd  F11=Next.error

Some errors cause error messages to be displayed for subsequent correct commands. For example, if you forget to put a period after a command, the next command line is incorrectly treated as a continuation of the first command.

Determining the Causes for Compile-Time Errors

When you attempt to add this process module to a dialog, compile errors will occur. You use the Dialog Process Source screen to determine the causes for compile-time errors. To do this, you:

1. **View the Compiler Process Modules screen** by selecting the **Display messages** option from the **Display** activity on the action bar of the Main Menu.

2. **View the Dialog Process Source screen** by selecting the **Display** activity from the Compiled Process Modules screen.

   Messages are displayed on this screen after statements that ADSC cannot compile. For example, assume that you made the following mistakes when you defined process module XXX DADD-RESPONSE earlier in this section:

   READY USAGE MODE UPDATE.
   IF AGR-CURRENT-RESPONSE NE SPACES
   AND FIELD DEPT-ID-0410 NOT CHANGED
   THEN
      EXECUTE NEXT FUNCTION.
   OBTAIN CALC DEPARTMENT
   IF DB-REC-NOT-FOUND
   THEN DO.
   STORE DEPARTMENT.
   DISPLAY MSG TEXT
   'DEPARTMENT ADDED'.
   END.

   DISPLAY MSG TEXT
   'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.

   - A dollar sign ($) is displayed below the first character in a statement in error.
The dash missing from the preceding USAGE-MODE keyword causes an error message.

The period missing from the OBTAIN command causes the subsequence IF statement to be in error.

The error in the above IF statement causes the error for the END statement.

In this example, the error in the USAGE-MODE keyword results in diagnostic messages for two correct commands that follow the OBTAIN command. You do not need to change these two correct commands.

3. **Page the screen back and forth** if necessary:

   - Press [PF8] to page forward.

4. **Note what requires correction** in the process module source or in other dialog components. It is a good idea to write down the errors that you note.

When you have looked at your errors, press [PF5] to go directly to IDD. The process module is question will be displayed on the screen.

**Correct structural errors**

At compile time, errors in a process module can arise because of discrepancies between the process module and other dialog components. These are called **structural errors**.

For example, you may have added the wrong subschema to the dialog. In this case, error messages are returned for process commands that reference records that exist in the correct subschema but don't exist in the specified subschema.

To correct discrepancies caused by other dialog components, you display the ADSC screen that associates the component with the dialog:

- For a **map**, display the Map Specifications screen.
- For a **subschema**, display the Database Specifications screen.
- For a **work record**, display the Records and Tables screen.

Dialog XXXDADD doesn't contain a work record, so you don't need to check for work-record errors in the dialog.

**Checking the Subschema Definition**

For example, you check the subschema definition for dialog XXXDADD by displaying the Database Specifications screen:

```
Database Specifications
Dialog XXXDADD  Version 1

Subschema . . . . . . . . . . . . EMPSS01
```
When checking a component specification, you need to verify that both the correct **name** and **version number** are specified.

For example, dialog XXXDADD uses subschema EMPSS01. This subschema is defined in version 1 of schema EMPSCHM. Some sites define different copies of schema EMPSCHM. In this case, you check the Database Specifications screen:

- **Expected specification**:
  
  SUBSCHEMA.: EMPSS01      SCHEMA: EMPSCHM    VERSION: 1

- **Incorrect specification**:
  
  SUBSCHEMA.: EMPSS01      SCHEMA: EMPSCHM    VERSION: 2

To correct any specifications, type the correct information over the previous specification. When dialog components are all specified correctly, you can proceed to correct any syntax errors in the process module.

### Correct syntax errors

IDD is used to define process modules and to correct syntax errors in the process modules. To transfer from ADSC to IDD, you press [PF5] from the Dialog Process Source screen.

**Note:** When you transfer to IDD, your current ADSC session is saved. You can transfer back to ADSC and resume the saved definition session when you are done with IDD.

The IDD screen will display the process module in question.

```
IDD 15.0 ONLINE      NO ERRORS      DICT=DEMO    1/4
MOD PROCESS XXXDADD-PREMAP
PROCESS SOURCE FOLLOWS
DISPLAY MSG TEXT
  'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
MSEND.
```

Correct the errors in the process module and press [Enter] to store the revised code.
After you correct all errors in a process module, you can return to the dialog definition in ADSC. To do this, enter **end** command area at the top of the IDD screen:

```
-> end
```

[Enter]

**Update dialogs that use the process module**

After you correct errors in a process module, you use ADSC to update the corrected process module in the dialog. When you do this, ADSC attempts to compile the source commands in the corrected process module. If the module compiles without errors, ADSC adds the process module to the dialog.

**Recompile the Dialog**

One way to update modified process modules in a dialog is to recompile the dialog:

```
Add Modify Compile Delete Display Switch
```

1. Compile
2. Display messages
---
international, Inc.
F3=Exit
---

Dialog name . . . . . . . XXXDADD
Dialog version . . . . . 1
Dictionary name . . . . . DEMO
Dictionary node . . . . . _______

Screen . . . . . . . . . 1 1. General options
2. Assign maps
3. Assign database
4. Assign records and tables
5. Assign process modules

Command ===> Enter F1=Help F3=Exit F10=Action

When you press [Enter], ADSC attempts to compile corrected process modules. If there are no errors:

1. ADSC adds the compiled module to the dialog.
2. ADSC recompiles the dialog load module.
3. ADSC displays the Main Menu screen with a confirming message

When the process module is successfully corrected and updated to the dialog, you can resume definition and testing procedures for the dialog.

**Test Process Logic**

To test process logic in dialog XXXDADD, you do the following:

- Invoke the Department Application (see page 318)
Invoke the Department Application

To invoke the Department application from DC/UCF, you enter the task code (XXXDEPT) for the application. For example, when using CA IDMS/DC, you invoke the Department application as shown:

ENTER NEXT TASK CODE:
xxxdept
[Enter]

For more information on invoking the Department application, see Instructions For Executing the Application (see page 250).

While testing dialog XXXDADD in this section, it is recommended that you write down the ID number of each sample department record that you add to the database. This will help you later in this section, when you need to locate, modify, and delete your sample department records.

From DEPTMENU, you can display the ADDDEP function and try out dialog XXXDADD:

x ADD
[Enter]

Enter sample information.

⚠️ Note: You must specify a unique ID number for each department. Try entering the last four digits of your home phone number.

Entering Sample Information

FUNCTION: ADDDEP
DATE....: 10/30/99

DEPARTMENT INFORMATION

DEPARTMENT ID .......: 9876
NAME .......: test department
HEAD ID ..: 1234

RESPONSE:

Display Function ADDDEP

The data is redisplayed after it is added.

FUNCTION: ADDDEP
DATE....: 10/30/99

DEPARTMENT INFORMATION

DEPARTMENT ID .......: 9876
RESPONSE:
DEPARTMENT ADDED

Notice the message you defined in response process **XXXDADD-RESPONSE**. It confirms that the redisplayed data has been added to the database.

When you press [Enter] to add a new department, response process **XXXDADD-RESPONSE** is executed. Statements in this response process test your input data and, for a new department, add the data to the sample database. The following diagram shows how components of dialog **XXXDADD** are executed at runtime when you use the dialog to add a department to the database. Dialog **XXXDADD** allows you to add new departments to the database. Execution remains in the dialog until the user requests another function (for example, MOD).

---

**IDMSDB--Executing the Application**

In dialog **XXXDADD**, you add a new department to the database by pressing [Enter]. To verify that your process logic keeps you from adding duplicate departments, try pressing [Enter] again to add the redisplayed department information.

After you press [Enter], the screen redisplays the department values along with an error message:
TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT

You defined this error message in process module XXXDADD-RESPONSE. The message is displayed whenever the specified department record already exists in the database. In this case, the duplicate record is not added to the database.

Test the XXXDADD Dialog

Try seeing what happens if you enter a new department record and also specify a valid response:

- Type new values over the redisplayed department information.
- Enter a valid response name (for example, MOD).

FUNCTION: ADDDEP
DATE....: 10/30/99

DEPARTMENT INFORMATION

DEPARTMENT ID . . . . . .......: 5432
   NAME ........: sample department
   HEAD ID ..: 1111

RESPONSE: mod

TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT

Notice the message displayed when you added TEST DEPARTMENT above does not affect your current add operation.

After you press [Enter], dialog XXXDADD's screen is redisplayed with the following message:

DEPARTMENT ADDED

Mode of Execution

Even when you specify a valid response, control remains in dialog XXXDADD when you add a new department. This type of execution is called STEP mode execution. In STEP mode, the screen is always redisplayed with a confirming message after a successful add, modify, or delete operation.

Dialog XXXDADD executes in STEP mode because of the following conditional command that you coded at the beginning of response process XXXDADD-RESPONSE:

IF AGR-CURRENT-RESPONSE NE SPACES   -- If the user specifies a valid response
   AND DEPT-ID-0415 NOT CHANGED        -- and doesn't enter a new department ID,
   THEN
      EXECUTE NEXT FUNCTION.            -- control transfers to the next function.

Dialogs do not have to execute in STEP mode. As an application developer, you can make use of the following execution strategies when developing an application:
Dialog response processes can execute in **FAST mode**. In this case, you write process logic that can handle necessary data operations and then transfer control without first redisplaying the screen. If the user inputs errors, the current screen can be redisplayed with an error message. This strategy benefits experienced users, who can input data and advance directly to the next function without any extra keystrokes.

Application functions can be **immediately executable**. In this case, the runtime system executes the function as soon as the user requests transfer to the function. Process logic is not executed before the transfer occurs. This strategy ensures that users do not inadvertently alter the database when transferring to another function. To make a function immediately executable, you use the ADSA Response Definition screen. For more information on immediately executable functions, see the *CA ADS Reference Section*.

**Another Test**

Now, test whether your process logic allows users to transfer control to other functions. According to your application design, for example, you can transfer to function MODDEP in either of the following ways:

- **You can press the control key** ([PF5]) associated with MODDEP.
- **You can enter the associated response name** (MOD) in the map's response field and press [Enter].

When you request transfer to MODDEP:

1. The runtime system stores the name (MOD) of the specified response in system-supplied element AGR-CURRENT-RESPONSE.
2. The runtime system executes process module XXXDADD-RESPONSE for the dialog, whether you pressed [Enter] or [PF5] to request transfer.

If you haven't already, transfer to MODDEP:

```
RESPONSE: mod
[Enter]
```

Since you didn't enter a department record along with the response name, control transferred to the indicated function. This processing was handled by the following conditional command at the beginning of process module XXXDADD-RESPONSE:

```
IF AGR-CURRENT-RESPONSE NE SPACES   -- You entered a valid response
AND DEPT-ID-0415 NOT CHANGED          -- and didn't enter a new department;
THEN
   EXECUTE NEXT FUNCTION.          -- therefore, control transferred.
```

**How Components are Executed**

The following diagram shows how components for dialog XXXDADD are executed at runtime when you request transfer to another application function. Dialog XXXDADD has a response process associated with [Enter]. No response process is associated with [PF5], the control key defined for the MOD response.
IDMSDB--Executing the Application (2)

It is unnecessary to test function MODDEP at this time since you have not yet defined any process logic for the associated dialog. You can exit from the Department application by specifying the EXIT response in the RESPONSE field:

RESPONSE: exit

[Enter]

Summary 5

In this section, you enhanced dialog XXXDADD so that it performs all operations necessary for final production use. You added processing logic to the dialog by performing the following steps:

1. **You defined two process modules** by using the IDD menu facility:

   - Process module XXXDADD-PREMAP displays the dialog’s map with a message for the user.
   - Process module XXXDADD-RESPONSE evaluates end-user input and stores department information in the database.

2. **You added the process modules to dialog XXXDADD** by using ADSC:

   - As a premap process, process module XXXDADD-PREMAP will be executed whenever dialog XXXDADD begins execution.
   - As a response process associated with [Enter],R process module XXXDADD-RESPONSE will be executed whenever the user presses [Enter] input data on the map for dialog XXXDADD.
3. **You executed the Department application** to see how the process modules affect execution of dialog XXXDADD:

- **You added new departments**, one at a time, by entering department information on the dialog’s map.
- **You transferred to function MODDEP** when you were finished using ADDDEP to add new departments.

Both developers and users can execute a dialog to test its process logic. Based on tests, developers and users often suggest modifications to process logic.

For example, after testing a fully developed dialog, users might find it confusing that the current screen is redisplayed if it contains input errors when the user enters BACK or EXIT. As an application developer, you might modify the application so that BACK and EXIT exit the user unconditionally from a dialog. To do this, you would use ADSA to modify BACK and EXIT, making the associated functions immediately executable.

As another example, users testing dialog XXXDADD might request that, after a new department is successfully added, the screen be redisplayed without the department’s information. This modification would make it more obvious that the redisplayed screen can be used to add another department.

In the next section, you will modify process module XXXDADD-RESPONSE so that it redисplays an initialized screen after a new department is added to the database.

### Modifying Process Logic in a Dialog

In the previous section, you defined a premap and a response process for dialog XXXDADD. You then executed the Department application to see how these processes affect dialog XXXDADD at runtime.

Based on runtime tests, it may be necessary to modify a dialog's premap or response process. This section provides instructions for modifying process module XXXDADD-RESPONSE, which you defined in the previous section. This section includes:

- An overview of modifying process logic
- Steps for modifying process modules
- Steps for updating modified process modules in dialogs
- A summary of what you've accomplished in this section

You can modify a dialog's premap or response process at any time in an application's life cycle. For example, assume that users who test out dialog XXXDADD ask you to change processing so that successfully added department data is not redisplayed to the user. To accomplish this change, you need to modify process module XXXDADD-RESPONSE, which evaluates, stores, and redisplays department data input by users.

**Initializing the Map**
To initialize map XXXMAP after new department data is added to the database, you will add an INITIALIZE command to process XXXDADD-RESPONSE. The modified process module is shown below.

```
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
    EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.

STORE DEPARTMENT.
    INITIALIZE (DEPARTMENT).  --- This command initializes the
    DISPLAY MSG TEXT            DEPARTMENT record buffer after the values
    'DEPARTMENT ADDED'.          in the buffer are stored in the database.
END.

DISPLAY MSG TEXT
    'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

When you modify process module XXXDADD-RESPONSE in this section, you will:

1. **Modify source commands for the process module** by using the IDD menu facility. You used the IDD menu facility in "Adding Process Logic to a Dialog", to define process module XXXDADD-RESPONSE.

2. **Update the modified process module in dialogs that use the process** by using the CA ADS catalog compiler (ADSC). In this section, you will use ADSC to update modified process module XXXDADD-RESPONSE in dialog XXXDADD.

Steps for modifying a process module and updating the modified process module in a dialog are presented below.

## Modifying Process Modules Using IDD

### Contents
- **Step 1 Retrieve the Process Module Definition** (see page 324)
- **Step 2 Modify Source Statements** (see page 325)

You can use IDD menu facility screens to modify a process module. In this section, you will modify process module XXXDADD-RESPONSE by performing the following steps:

1. Retrieve the process module definition.
2. Modify process module source statements.

These steps are discussed below.

### Step 1 Retrieve the Process Module Definition

In order to retrieve a process module, you must first invoke the IDD menu facility by using its task code (for example, IDDMT). For example, when using CA IDMS/DC, you invoke the IDD menu facility as shown:
ENTER NEXT TASK CODE:
iddmt

After you invoke the IDD menu facility, you may need to provide signon information.

For more information on invoking and signing on to the IDD menu facility, see .

You display a process module definition on the Process Entity screen. You can transfer to this screen and request display of an existing process module at the same time. To do this, enter the identifier for the screen (PROC) in the command area of an IDD menu facility screen, followed by the name of the process module.

Displaying a Process Module Definition

For example, you can display process module XXXDADD-RESPONSE from the Master Selection screen by specifying PROC and the process module name in the command area.

CA CAGJF0
IDD REL 15.0 *** MASTER SELECTION *** TOP
-> proc xxdadd-response

DICTIONARY NAME...: DEMO
NODE NAME...:
USER NAME.........:
PASSWORD.........:
USAGE MODE........: X UPDATE _ RETRIEVAL
PFKEY SIMULATION..: X OFF _ ON

The process module definition is displayed.

Process Entity Screen

-> IDD REL 15.0 *** PROCESS ENTITY *** PROC
   DICT=DEMO
   X DISPLAY PROCESS NAME....: XXXDADD-RESPONSE
   MODIFY _ ADD
   _ DELETE VERSION NUMBER..: 1 _ HIGHEST _ NEXT HIGHEST
   _ LOWEST _ NEXT LOWEST
   DESCRIPTION......:

After you press [Enter], IDD displays basic information on the specified process module on the Process Entity screen, along with a message like:

PROCESS 'XXXDADD-RESPONSE' VERSION 1 DISPLAYED

You can make modifications to specifications on the Process Entity screen, if necessary, and then press [Enter] to store the modified information.

Step 2 Modify Source Statements

You defined process module XXXDADD-RESPONSE in . In this step, you will modify source commands in the process module so that the dialog's screen is initialized for redisplay after the user inputs a new department record.
To modify source commands for a process module, you use the Process Source screen. You can display the Process Source screen for process module XXXDADD-RESPONSE as shown.

Displaying the Process Source Screen

Type the screen identifier (SRCE) in the command area.

-> srce [Enter]

The Process Source screen is displayed.

```plaintext
PROCESS 'XXXDADD-RESPONSE' VERSION 1
---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
  EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.
  STORE DEPARTMENT.
  DISPLAY MSG TEXT
    'DEPARTMENT ADDED'.
END.

DISPLAY MSG TEXT
  'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

Modifying the Process Module

To modify process module XXXDADD-RESPONSE, you need to insert an INITIALIZE RECORDS command in the source commands for the process module:

1. Place the cursor on the line after which new statements are to be added.

2. Press the control key that inserts new lines on IDD screens (a different control key may be defined at your site).
DISPLAY MSG TEXT
'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.

Add the INITIALIZE statement and press the control key that applies changed or inserted lines to IDD screens (a different key may be defined at your site).

IDD REL 15.0 *** PROCESS SOURCE *** SRCE
-> PROCESS 'XXDADD-RESPONSE' VERSION 1
---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
STORE DEPARTMENT.
initialize (department).

The updated process statements are displayed.

Press [Enter] to store the updated process module in the data dictionary:

READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
  EXECUTE NEXT FUNCTION.

STORE DEPARTMENT.
INITIALIZE (DEPARTMENT).
DISPLAY MSG TEXT 'DEPARTMENT ADDED'.

DISPLAY MSG TEXT
'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.

When you press [Enter], IDD updates the process module in the data dictionary. Then, the Process Source screen displays a message like:

PROCESS 'XXDADD-RESPONSE' VERSION 1 MODIFIED

>> For more information on inserting lines in a process module, see CA IDMS Common Facilities Section.

Exit from IDD

After you successfully modify process module XXXDADD-RESPONSE, you can exit from IDD. In this sample session, you will transfer directly from IDD to ADSC to update dialog XXXDADD. To do this, use the SWITCH command, followed by the task code for ADSC.

You enter the SWITCH command in the command area of any IDD menu facility screen:

-> switch adsct

After you exit from IDD, you can use ADSC to update the modified process module in any dialog that uses it.
Updating Modified Process Modules In Dialogs Using ADSC

Contents

- Step 1 Retrieve and Check Out the Dialog (see page 328)
- Step 2 Recompile the Dialog (see page 330)
- Execute the Application (see page 330)

In the above step, you used the IDD menu facility to modify process module XXXDADD-RESPONSE. Now, you will recompile dialog XXXDADD, which uses this process module. When you recompile the dialog, ADSC compiles the modified source statements for the process module.

You will perform the following steps:

1. Retrieve and check out the dialog to be updated.
2. Recompile the dialog.

After you recompile the dialog and exit from ADSC, you can execute the application to test the modified dialog.

Step 1 Retrieve and Check Out the Dialog

In order to retrieve a dialog definition, you must be using ADSC.

If you did not transfer directly to ADSC earlier in this section when you exited from IDD, you need to invoke ADSC by using the task code (for example, ADSCT) for ADSC.

ADSC begins by displaying the Main Menu screen. You use the Main Menu screen to retrieve a dialog definition for update.

Retrieving the Dialog

For example, you can retrieve dialog XXXDADD:

Add Modify Compile Delete Display Switch

CA ADS Online Dialog Compiler
CA, Inc.

Dialog name . . . . . . . xxxdadd
Dialog version . . . . . 1
Dictionary name . . . . . demo
Dictionary node . . . . . ________
Screen . . . . . . . . . . 1 1. General options
2. Assign maps
3. Assign database
4. Assign records and tables
5. Assign process modules

Copyright (C) 2003 CA, Inc.
Checking Out an Application

To check the application out for modification, position the cursor on the Modify item on the action bar and press [Enter]. You can position the cursor on Modify by:

- Pressing [PF10] to move to the action bar and then tabbing to Modify and pressing [Enter]
- Tabbing to Modify and pressing [Enter]
- Typing modify on the command line and pressing [Enter]

Add  Modify  Compile  Delete  Display  Switch

Press [Enter] to check the dialog out.

Note: If the dialog has been checked out to another developer and has not been either compiled or released by that developer, you will not be able to check it out. The other developer must release the dialog.

After you press [Enter] to retrieve the dialog, ADSC displays basic information on dialog XXXDADD on the Main Menu screen along with the following message:

NO ERRORS DETECTED

After you successfully check out XXXDADD, you can update the modified process module in the dialog.
Step 2 Recompile the Dialog

Earlier in this section, you modified source statements for process module XXXDADD-RESPONSE. In order to include the modified process module in dialog XXXDADD, you must recompile the dialog. When you recompile the dialog, the modified source module is compiled. If the compilation is successful, the compiled process module is included in the dialog load module.

To recompile dialog XXXDADD, select the **Compile** activity from the action bar on the Main Menu of the dialog compiler.

```
Add Modify Compile Delete Display Switch

1 1. Compile 2. Display messages
              log Compiler
   ------------------------------------
   Dialog name . . . . . . . . XXXDADD
   Dialog version . . . . 1
   Dictionary name . . . . DEMO
   Dictionary node . . . .
   Screen . . . . . . . . 1 1. General options
                        2. Assign maps
                        3. Assign database
                        4. Assign records and tables
                        5. Assign process modules

Command ===> Enter F1=Help F3=Exit F10=Action
```

After you press [Enter], ADSC displays either:

- A confirming message when process modules contain no errors and the dialog load module has been successfully recompiled.
- An error message when errors in a process module prevent ADSC from successfully recompiling the dialog.
  In this case, you correct errors in the process module as described in .

After you successfully recompile the dialog, you can exit from ADSC by pressing [PF3].

Execute the Application

You can now execute the Department application to see how adding an INITIALIZE RECORDS command to process module XXXDADD-RESPONSE affects dialog XXXDADD at runtime.

You can invoke the Department application from DC/UCF by entering the task code (for example, XXX DEPT) for the application. For example, when using CA IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE: xxxdept
[Enter]
```

According to your application design, ADDDEP is the only function that executes dialog XXXDADD.

The ADDDEP function allows you to add new department information. When you used the ADDDEP function to add sample departments in , your sample data was redisplayed to you after you added the data to the application database by pressing [Enter].

Entering Sample Information

Display ADDDEP and then try adding a sample department now that you have modified process module XXXDADD-RESPONSE.

⚠️ Note: Enter different department data than shown here.

FUNCTION: ADDDEP
DATE....: 10/30/99
DEPARTMENT INFORMATION

DEPARTMENT ID ........: 1098
NAME ........: sample department
HEAD ID ..: 1098

RESPONSE:
Screen is Redisplayed

An initialized screen is displayed after you add new department data.

FUNCTION: ADDDEP
DATE....: 10/30/99
DEPARTMENT INFORMATION

DEPARTMENT ID ........:
NAME ........:
HEAD ID ..:

RESPONSE:
DEPARTMENT ADDED

When you are finished testing the ADDDEP function, you can exit from the Department application by entering the EXIT response in the screen's RESPONSE field.

Summary 6

In this and previous sections, you developed dialog XXXDADD to add new departments to the database for the Department application by performing the following steps:
1. **You defined dialog XXXDADD as a skeleton dialog** for the prototype Department application in Section10, "Defining Dialogs Using ADSC" by adding map XXXMAP to the dialog.

2. **You enhanced the dialog to full production capability** in Section13, "Adding Process Logic to a Dialog", by adding a subschema, premap process, and response process to the dialog.

3. **You completed the dialog** in this section by modifying the dialog's response process based on end-user testing of the dialog.

Now that you have completed dialog XXXDADD, you can proceed to enhance dialog XXXDUPD by defining a work record and modules of process code for the dialog.

---

### Defining Work Records Using IDD

In previous sections, you used the IDD menu facility to define and modify process modules for use in dialog XXXDADD. You also can use IDD to define work records for use by dialogs.

Instructions for defining work records are provided in this section. The sample work record defined in this section will be used by process modules that you will define for dialog XXXDUPD in Section16, "Completing the Department Application". This section includes:

- An overview of defining work records for CA ADS applications
- Instructions for defining work records
- A summary of what you've accomplished in this section

⚠️ **Note:** At some sites, all records and elements are defined by database administrators (DBAs). Even if this is the case at your site, go ahead and read this section to find out more about records in the CA ADS environment.

Each item of data used and stored by your application must be defined as an **element** in the data dictionary. Most elements describe database values. For example, DEPT-ID-0410, DEPT-NAME-0410, and DEPT-HEAD-ID-0410 are elements that store information on a department in the database.

Elements are grouped together in **records**. For example, elements DEPT-ID-0410, DEPT-NAME-0410, and DEPT-HEAD-ID-0410 are grouped together in the DEPARTMENT record because they all store information on a department. The DEPARTMENT record is called a **database record** because it describes information in the database. Database records are defined by using the schema compiler. (Alternatively, in an SQL environment, tables are defined by using SQL.) DBAs usually are responsible for defining and maintaining database records.

You also can define **work records** for an application. Work records describe temporary storage for a dialog. Elements in a work record describe data that is not stored in the application database. You define work records by using the **IDD menu facility**.

⚠️
Note: Work records are sometimes referred to as IDD records to differentiate them from schema-defined database records.

Database and work records must be associated with each dialog that uses them.

**Associating Database Records with a Dialog**

You associate database records with a dialog when you add a subschema containing the records to the dialog.

For example, subschema EMPSS01 contains the DEPARTMENT record. When you added subschema EMPSS01 to dialog XXXDADD in , you made the DEPARTMENT record available to that dialog.

**Associating Work Records with a Dialog**

You associate work records with a dialog either by naming the record when you define the dialog's map or when you define the dialog itself, depending on how the work record is to be used:

- **If the work record is to be displayed on the dialog's map** (and optionally used in the dialog's process logic), you add the record with the map. Work records named on a map are automatically available to dialogs that use the map.

- **If the work record is to be used only by process module commands** (and not displayed on the dialog's map), you add the work record directly to the dialog.

Note that both database and work records that define data to be displayed on a map must be directly associated with the map. Because of this, records on a map are often referred to collectively as map records.

**Note:** Work records associated with a map are automatically available to dialogs that use the map. For database records associated with a map, you still must give the dialog a subschema to make the records available to the dialog. This is because information on the database itself is defined in the subschema, and is required by dialogs that access database records.

In this section, you will define a work record for dialog XXXDUPD. Data for the record will be used internally by process modules but will not be displayed to the end user. Therefore, you will associate this record directly with dialog XXXDUPD (rather than with map XXXMAP) when you add process logic to the dialog in Section16, "Completing the Department Application".

**Instructions 3**

**Contents**

- Step 1 Define an Element (see page 334)
- Step 2 Define a Work Record (see page 337)
- Step 3 Specifying Basic Information (see page 337)
Adding Elements (see page 338)

Work records typically are small and contain only elements needed by a particular dialog or process module. This reduces the amount of storage reserved for a given dialog during execution. For example, in this section you will define a one-element work record for dialog XXXDUPD.

You will use the **IDD menu facility** to define:

- **Element XXX-WK-FIRST-TIME** -- An element for use in response process XXXDUPD-ENTER. Element XXX-WK-FIRST-TIME stores either Y (yes) or N (no) to indicate whether response process XXXDUPD-ENTER is processing a department record for the first time, or not. Different commands in process module XXXDUPD-ENTER are performed, based on the value in element XXX-WK-FIRST-TIME.

- **Record XXX-WK-RECORD** -- A work record for dialog XXXDUPD. Record XXX-WK-RECORD contains element XXX-WK-FIRST-TIME.

Additional elements can be added to a work record at any time. For example, if a response process for dialog XXXDUPD is modified so that a counter is required, you can add an element (for example, XXX-WK-COUNTER) to record XXX-WK-RECORD to store the counter value.

To define a work record in this section, you will:

1. Define element XXX-WK-FIRST-TIME.
2. Define work record XXX-WK-RECORD.

These steps are presented below. If you need additional information at any time about the use of IDD, see .

**Step 1 Define an Element**

Each **element** that you add to the data dictionary describes a unit of data that can be used in an application.

In order to define an element, you must invoke and sign on to the IDD menu facility. For example, when using IDMS-DC, you invoke the IDD menu facility as shown:

```plaintext
ENTER NEXT TASK CODE:
iddmt
[Enter]
```

For more information on invoking and signing on to the IDD menu facility, see .

You use the **Element Entity screen** to define elements in the data dictionary. You display the Element Entity screen as shown:

```plaintext
-> elem
```

**Element Entity Screen**

```plaintext
-> IDD REL 15.0
*** ELEMENT ENTITY ***
ELEM
dict=DEMO
```
X DISPLAY ELEMENT NAME.....:
  _ MODIFY
  _ ADD VERSION NUMBER...: 1 _ HIGHEST _ NEXT HIGHEST
  _ DELETE _ LOWEST _ NEXT LOWEST
DESCRIPTION:
PICTURE.....: NO SYNC: X _ SYNC: _

USAGE......: X DISPLAY _ CONDITION NAME (LEVEL 88)
  _ COMP/COMP-4 (BINARY) _ COMP-3 (PACKED DECIMAL)
  _ COMP-1 (SHORT FLOATING) _ COMP-2 (LONG FLOATING)
  _ BIT _ POINTER
  _ ELMX = ELEMENT EXTENSION <PF9> _ SUBE = SUBORD ELEMENTS <PF11>
  _ REGN = USER REGISTRATION <PF2> _ PUBL = PUBLIC ACCESS <PF3>
  _ CLAT = CLASS/ATTRIBUTES <PF4> _ RKEY = RELATIONAL KEYS <PF5>
  _ COMM = COMMENTS <PF6> _ COML = COMMENT KEY LIST <PF7>
  _ HIST = HISTORY <PF8> _ COPY = SAME AS/COPY FROM
  _ XREF = CROSS REFERENCE <PF10> _ HELP = HELP <PF1>

Screen Prompts

When you use the Element Entity screen to define an element, you typically enter specifications for the following Element Entity screen prompts:

- **ELEMENT NAME** -- You must supply an element name. The name you specify must be unique.

- **DISPLAY** -- You deselect the DISPLAY action when you intend to add a new element.
  To do this, type a blank over the X displayed to the left of the action.

- **ADD** -- You must select the ADD action to specify that you are defining a new element.

- **DESCRIPTION** -- You optionally specify a brief description of the element.

- **PICTURE** -- You must specify the storage layout (that is, picture) for the element after the PICTURE prompt. A picture specifies:
  - The **type of data** that can be stored for the element:
    - For **alphanumeric** data, enter an X.
    - For **alphabetic** data, enter an A.
    - For **numeric** data, enter a 9.
  - The **number of characters** that can be stored for the element.
    By default, an element can store **single-character values**.
    You optionally enable an element to store **larger values** by specifying, in parentheses, the maximum number of characters to be stored by the element. For example, a picture of X(6) enables an element to store values that contain up to six alphanumeric characters.
    For example, the element that you define in this section will be used to store the single-character value Y or N.
• **USAGE** -- You optionally specify the storage format for data after the USAGE prompt.
  For element *XXX-WK-FIRST-TIME*, you will retain the default usage, **DISPLAY**, since **DISPLAY** usage is appropriate for the flag values (Y and N) stored for the element.
  For more information on **DISPLAY** and other USAGE specifications, see the *CA IDMS IDD Quick Reference Section*.

**Defining an Element**

To define *XXX-WK-FIRST-TIME* as an element that stores single-character alphanumeric values, you use the Element Entity screen to:

- Type the name of the element. You can use your initials instead of *XXX*
- Deselect the **DISPLAY** action and select the **ADD** action.
- Type an optional description for the element.
- Specify a picture of X. This enables the element to store single-character alphanumeric flag values.

```
IDD REL 15.0 *** ELEMENT ENTITY *** ELEM
->
    DICT=DEMO

    DISPLAY ELEMENT NAME.....: xxx-wk-first-time
    MODIFY
    x ADD VERSION NUMBER....: 1 HIGHEST NEXT HIGHEST
    _ DELETE LOWEST NEXT LOWEST

    DESCRIPTION: passes flag value for dialog xxxdupd

    PICTURE.....: x NO SYNC: X SYNC: _

    USAGE.......: X DISPLAY _ CONDITION NAME (LEVEL 88)
    _ COMP/COMP-4 (BINARY) COMP-3 (PACKED DECIMAL)
    _ COMP-1 (SHORT FLOATING) COMP-2 (LONG FLOATING)
    _ BIT _ POINTER
    _ ELMX = ELEMENT EXTENSION <PF9> _ SUBE = SUBORD ELEMENTS <PF11>
    _ REGN = USER REGISTRATION <PF2> _ PUBL = PUBLIC ACCESS <PF3>
    _ CLAT = CLASS/ATTRIBUTES <PF4> _ RKEY = RELATIONAL KEYS <PF5>
    _ COMM = COMMENTS <PF6> _ COML = COMMENT KEY LIST <PF7>
    _ HIST = HISTORY <PF8> _ COPY = SAME AS/COPY FROM
    _ XREF = CROSS REFERENCE <PF10> _ HELP = HELP <PF1>
```

After you press [Enter], the Element Entity screen displays a message to indicate whether the element definition has been added to the data dictionary:

- **If you successfully defined a new element**, IDD displays a message like:
  
  ELEMENT ‘XXX-WK-FIRST-TIME’ VERSION 1 ADDED

- **If the element cannot be added to the data dictionary**, IDD displays a different message.
  Read the message to determine the problem. You can type over any errors and press [Enter] again.

After you successfully add element *XXX-WK-FIRST-TIME* to the data dictionary, you can define work record *XXX-WK-RECORD*. 

---

*CA IDMS - 19.0*

13-Jan-2018 336/442
Step 2 Define a Work Record

You define a record to describe a collection of one or more existing elements. To define a work record to the data dictionary, you perform the following steps using the IDD menu facility:

1. **You specify basic information for the record** by using the Record Entity screen.

2. **You add one or more existing elements to the record** by using the Record Element screen. In this section, you will use this screen to add element XXX-WK-FIRST-TIME to record XXX-WK-RECORD.

The way you specify basic information for a work record and add elements to the record are shown below.

Step 3 Specifying Basic Information

You use the **Record Entity** screen to define a work record. You display the Record Entity screen as shown:

-> recd

[Enter]

**Record Entity Screen**

The **Record Entity** screen is displayed.

-> IDD REL 15.0 *** RECORD ENTITY *** RECD

[IDT=DEMO]

**Screen Prompts**

When adding a new work record, you usually specify information for the following Record Entity screen prompts:

- **RECORD NAME** -- You must supply a record name. The name that you specify must be unique.

- **DISPLAY** -- You deselect the DISPLAY action when you intend to add a new record.
ADD -- You select the ADD action to specify that you are defining a new record.

DESCRIPTION -- You optionally provide a brief description of the record.

To specify basic information for record XXX-WK-RECORD. You use the Record Entity screen and enter the indicated specifications:

- Type the name of the record You can use your initials instead of XXX
- Deselect the DISPLAY action and select the ADD action.
- Type an optional description for the record

```
-> IDD REL 15.0            *** RECORD ENTITY ***            RECD
                   DICT=DEMO

DISPLAY            RECORD NAME.....: xxx-wk-record

MODIFY
  ADD                VERSION NUMBER..: 1        HIGHEST     NEXT HIGHEST
                    _ LOWEST     _ NEXT LOWEST

DESCRIPTION.....: work record for dialog xxdupd

RECORD LENGTH....:
```

After you press [Enter], the Record Entity screen displays a message to inform you whether the record definition has been stored in the data dictionary:

- If you successfully defined a new record, IDD displays a message like:

   RECORD 'XXX-WK-RECORD' VERSION 1 ADDED

- If the record cannot be added to the data dictionary, IDD displays a different message. In this case, read the message to determine the problem. You can change specifications on the Record Entity screen and press [Enter] again.

After you successfully specify basic information for a record, you can add elements to the record by using the Record Element screen.

Adding Elements

You defined element XXX-WK-FIRST-TIME in the data dictionary in Step 1: Define an Element (see page ) earlier in this section. In this step, you will add element XXX-WK-FIRST-TIME to record XXX-WK-RECORD. Elements used in records are referred to as record elements.

You use the IDD Record Element screen to add elements to work records:
RELM identifies the Record Element screen.

The Record Element screen is displayed.

```
RECORD 'XXX-WK-RECORD' VERSION 1
_ REMOVE
_ REPLACE
_ REDefines
_ OCCURS
_ DepENDING ON
_ PICTURE
_ USAGE
_ VALUE(S)
_ EXCLUDE VALUES

ELEMENT SYNONYM NAME.................: 
FOR RECORD SYNONYM...................: _ HIGHEST _ LOWEST
VERSION NUMBER...................: _ HIGHEST _ LOWEST
```

Screen Prompts

When you add an element to a record, you typically specify the following information:

- **Element name** -- You must name an existing element after the RECORD ELEMENT NAME prompt.
- **Additional specifications** -- You optionally can redefine how the element is used in the current record by using prompts on the Record Element screen. For example, you could override the PICTURE or USAGE specifications you gave to element XXX-WK-FIRST-TIME in Step 1: Define an Element (see page ) earlier in this section.
  
  You most often override specifications for an element when the element is used in different ways by several work records. Specifications made on the Record Element screen do not alter the actual element definition in the data dictionary. For more information on Record Element screen specifications, see the CA IDMS IDD Quick Reference Section.

  You will not change any values when you add element XXX-WK-FIRST-TIME to record XXX-WK-RECORD in this section.

You use the Record Element screen to type in the name of the element that you defined earlier in this section.
DEPENDING ON:

PICTURE:

USAGE:

VALUE(S):

NO SYNC
SYNC
DISPLAY
CONDITION NAME
BIT
POINTER
COMP
COMP-1
COMP-2
COMP-3

THRU
THRU
THRU

ELEMENT SYNONYM NAME:

FOR RECORD SYNONYM:

After you press [Enter], the Record Element screen displays a message to indicate whether the element has been added to the work record in the data dictionary:

- If the element is successfully added to the record, IDD displays a message like:

  RECORD 'XXX-WK-RECORD' VERSION 1 MODIFIED

- If the element cannot be added to the record, IDD displays a different message than the one shown above.
  In this case, read the message to determine the problem. You can type over any errors and press [Enter] again.

After you finish defining work elements and records, you can exit from IDD. In this sample session, exiting from IDD is optional because you will use the IDD menu facility again in the next section.

If you want to exit from IDD, enter the SWITCH SUSPEND command in the command area of any IDD menu facility screen:

  -> switch suspend

  [Enter]

Summary 7

You can define a work record to establish temporary storage for a dialog. Elements in the work record describe the data to be stored.

In this section, you used the IDD menu facility to create a work record for use in dialog XXXDUPD by performing the following steps:

1. You defined element XXX-WK-FIRST-TIME to store a single-character, alphanumeric status value. When used in dialog XXXDUPD, this element contains either Y (yes) or N (no). This value establishes whether response process XXXDUPD-ENTER is processing a department for the first time or not.

2. You defined record XXX-WK-RECORD to contain element XXX-WK-FIRST-TIME.

To see how you use work records and elements in dialogs, proceed to the next section, section 10, where you enhance dialog XXXDUPD by adding process modules and data definitions to the dialog.
Completing the Department Application

When you created the prototype of the Department application, you defined skeleton dialogs XXXDADD and XXXDUPD to be displayed by dialog functions in the application. In previous sections, you completed dialog XXXDADD by adding modules of process commands for the dialog.

In this section, you will complete the sample Department application by adding process logic to dialog XXXDUPD. The process modules that you define for the dialog will allow the end user to modify and delete existing department information.

This section includes:

- An overview of process modules and dialog execution
- Steps for defining process modules
- Steps for associating process modules with dialogs
- Steps for executing the application
- A summary of what you've accomplished in this section

The final structure of the sample Department application is shown below. Dialog XXXDUPD is the only component in the application that requires further development. In this section, you will enhance dialog XXXDUPD by writing three process modules for the dialog. When you complete dialog XXXDUPD, you will have finished defining the sample Department application. Dialog XXXDUPD, which is executed by functions MODDEP and DELDEP, is the only component in the sample application that requires further development.
As the diagram indicates, dialog XXXDUPD is executed by both functions MODDEP and DELDEP. Therefore, process modules that you write for dialog XXXDUPD must be able to both modify and delete department information in the database.

To handle all processing requirements for dialog XXXDUPD, you will define the following premap and response processes for the dialog:

<table>
<thead>
<tr>
<th>Process module</th>
<th>Type</th>
<th>Function performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXDUPD- PREMAP</td>
<td>Premap</td>
<td>Displays the dialog's map with a message prompting the end user for the department to be modified or deleted.</td>
</tr>
<tr>
<td>XXXDUPD- ENTER</td>
<td>Response (associated with the [Enter] key)</td>
<td>Handles most modification and deletion operations.</td>
</tr>
<tr>
<td>XXXDUPD- PA2</td>
<td>Response (associated with the [PA2] key)</td>
<td>Allows the end user to cancel the current modification or deletion operation before the database is updated.</td>
</tr>
</tbody>
</table>

When you add these process modules to dialog XXXDUPD, you also must enable the process modules to access temporary storage and database information. To do this, you will associate with the dialog any records that define temporary storage and database records. You will add:
1. **Work record XXX-WK-RECORD** -- Includes element XXX-WK-FIRST-TIME, which stores a status value (Y or N) used by response processes in dialog XXXDUPD.
   
   You defined work record XXX-WK-RECORD in num=14. Defining Work Records Using IDD (see page 332).

2. **Subschema EMPSS01** -- Includes the DEPARTMENT record, which is the database record for department information.
   
   At run time, the dialog can access any database record in the dialog's subschema. Database administrators (DBAs) typically define subschemas for use by dialogs.

The following diagram shows dialog XXXDUPD with all of its components. To enable dialog XXXDUPD to perform all necessary processing, you will add a premap process, two response processes, a work record, and a subschema to the dialog.

**IDMSDB--Completing the Department Application (2)**

**Cancelling a Modification**

To protect data in the database, your process modules should make it easy for end users to cancel a modification or deletion operation. As an application developer, you will do this by defining a two-stage procedure for modifying or deleting department records:

1. **First, the end user specifies a department to modify or delete.** When the end user presses [Enter] to specify a department, response process XXXDUPD-ENTE is executed. This response process accesses and displays the complete department record.

2. **Then, the end user either cancels or continues the current operation**, based on the record displayed on the screen:
- **To cancel**, the end user presses [PA2].

- **To continue**, the end user types any necessary modifications to the record and presses [Enter]. This time, response process XXXDUPD-ENTER modifies or deletes the department record in the database.

Response process XXXDUPD-ENTER performs both stages of a completed modification or deletion operation. This is accomplished at run time by having the status value (Y or N) in element XXX-WK-FIRST-TIME determine which stage is performed. Commands in your process modules change the value in XXX-WK-FIRST-TIME, as appropriate.

Instructions for defining process modules, adding the modules to a dialog, and executing the final application are given on the following pages.

**Define Process Modules Using IDD**

According to your application definition, dialog XXXDUPD is executed by functions MODDEP and DELDEP:

- Function **MODDEP** executes dialog XXXDUPD to modify department records.

- Function **DELDEP** executes dialog XXXDUPD to delete department records.

**Step 1 Define Process Module XXXDUPD-PREMAP (see page 345)**

**Step 2 Define Process Module XXXDUPD-ENTER (see page 347)**

**Step 3 Define Process Module XXXDUPD-PA2 (see page 353)**

Process modules that you write must be able to handle both modification and deletion operations. To accomplish this, you will include commands that:

1. Test which function (MODDEP or DELDEP) is currently in use

2. Invoke subroutines appropriate to the current function

You test which function is currently in use by querying **AGR-CURRENT-FUNCTION**, which is an element in the system-supplied ADSO-APPLICATION-GLOBAL-RECORD. At run time, AGR-CURRENT-FUNCTION stores the name of the current function. Your process module can access AGR-CURRENT-FUNCTION because ADSO-APPLICATION-GLOBAL-RECORD belongs to the dialog's map (and thus to the dialog).

For example, the following conditional command tests the value in AGR-CURRENT-FUNCTION:

```plaintext
IF AGR-CURRENT-FUNCTION EQ 'MODDEP'    -- If function MODDEP is executing
THEN
    CALL MODRTN.                        -- call subroutine MODRTN to
                                                     modify the department record.
```

In this section, you will define process modules for dialog XXXDUPD. You will:

1. Define process module XXXDUPD-PREMAP.

2. Define process module XXXDUPD-ENTER.
3. Define process module XXXDUPD-PA2.

**Step 1 Define Process Module XXXDUPD-PREMAP**

In order to define a process module, you must be using IDD. If you did not remain signed on to IDD at the end of the previous section, you should now sign on to the IDD menu facility, as described in Step 1: Invoke the IDD Menu Facility (see page 291).

The first process module you will define in this section is process module XXXDUPD-PREMAP. Statements that you will input for the process module are shown below. This process module will be the premap process for dialog XXXDUPD.

```
MOVE 'Y' TO XXX-WK-FIRST-TIME.           1
IF AGR-CURRENT-FUNCTION EQ 'MODDEP'     -
  THEN                                 2
    DISPLAY MSG TEXT
      'MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.-
  ELSE                                 -
    DISPLAY MSG TEXT
      'DELETE -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.-
```

1 This statement sets the flag value in XXX-WK-FIRST-TIME to Y (YES).

2 If the MODDEP function is in use, the dialog's map is displayed with the MODIFY message (defined here between the single quotation marks).

3 If the DELDEP function is in use, the dialog's map is displayed with the DELETE message.

**Specifying Basic Information About the Process Module**

You use the Process Entity screen to specify basic information for a process module. You can display and use the Process Entity screen as shown:

```
CA CAGJF0
IDD REL 15.0 *** MASTER SELECTION *** TOP
-> proc
SIGNON TO IDD WAS SUCCESSFUL
DICTIONARY NAME...: DEMO NODE NAME.: [Enter]
```

Enter the indicated specifications:

- Type the name of the process module. You can use your initials instead of XXX.
- Deselect the DISPLAY action and select the ADD action.
- Optionally type a description of the process.
DISPLAY PROCESS NAME.....: xxdupd-premap
MODIFY
X ADD VERSION NUMBER...: 1 _HIGHEST _NEXT HIGHEST
_ DELETE _LOWEST _NEXT LOWEST
DESCRIPTION.....: display map to mod/del departments

When you press [Enter], IDD redisplays the Process Entity screen with a message. If the process module is defined successfully, the Process Entity screen displays a message like:

PROCESS 'XXDUPD-PREMAP' VERSION 1 ADDED

If a different message is displayed, read the message to determine the problem. You can type over any errors, and then press [Enter] again.

After you specify basic information on a process module, you can use the Process Source screen to enter process commands for the process module. For example, enter process commands for process module XXDUPD-PREMAP as shown:

-> srcs

SRCE identifies the Process Source screen.

The Process Source screen is displayed.

IDD REL 15.0 *** PROCESS SOURCE *** SRCE
-> NO DATA LINES CURRENTLY EXIST

PROCESS 'XXDUPD-PREMAP' VERSION 1
---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

Entering Process Statements

Enter source statements for the process module:

- Enter keywords, periods, and single quotes as shown.
- The exclamation point in any column signals the start of a comment.
- You can types spaces to indent statements, making the process source easier to read and debug.

⚠️ **Caution:** Don't type any characters beyond column 72.

IDD REL 15.0 *** PROCESS SOURCE *** SRCE
-> NO DATA LINES CURRENTLY EXIST

PROCESS 'XXDUPD-PREMAP' VERSION 1
---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
mov 'y' to 'xxx-wk-first-time'.
! if agr-current-function eq 'moddep'
then
  display msg text
'modify -- enter the department id, or select: back or exit'.
!
else
    display msg text
    'delete -- enter the department id, or select: back or exit'.

[Enter]

The Process Source screen is redisplayed.

IDD REL 15.0             *** PROCESS SOURCE ***              SRCE
->                     1/10

PROCESS 'XXXDUPD-PREMAP' VERSION 1

MOVE 'Y' TO XXX-WK-FIRST-TIME
!
IF AGR-CURRENT-FUNCTION EQ 'MODDEP'
    THEN
        DISPLAY MSG TEXT
        'MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.
    !
ELSE
    DISPLAY MSG TEXT
    'DELETE -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.

After you press [Enter], IDD adds the process module to the data dictionary and redisplays the
Process Source screen with a message like:

PROCESS 'XXXDUPD-PREMAP' VERSION 1 MODIFIED

After you finish using the Process Source screen for process module XXXDUPD-PREMAP, you can
proceed to define process module XXXDUPD-ENTER.

Step 2 Define Process Module XXXDUPD-ENTER

Now you will define process module XXXDUPD-ENTER for dialog XXXDUPD. This process module will
be executed when the end user presses [Enter].

Process XXXDUPD-ENTER allows an end user to specify a department, verify that the correct
department is specified, and then enter modification or deletion instructions by performing the
following processing:

1. **When the end user first specifies the department to be processed and presses [Enter]**, response process XXXDUPD-ENTER:
   - Retrieves information stored for the specified department
   - Displays the department information for verification and prompts the end user to modify
     or delete the displayed record

2. **When the end user presses [Enter] again**, response process XXXDUPD-ENTER:
   - Modifies or deletes the department in the database
   - Displays the dialog’s map to the end user with a message confirming the modification or
     deletion operation

To enable process module XXXDUPD-ENTER to perform the above processing, commands in the
process module store and evaluate a status value in element XXX-WK-FIRST-TIME:
- **A value of Y (yes)** indicates that the process module is executing for the first time for a given department, and so must retrieve the department from the database. Before redisplaying the screen to the end user, the response process changes the value in element XXX-WK-FIRST-TIME from Y to N.

- **A value of N (no)** indicates that the process module is executing for the second time for the department, and so must modify or delete information stored for the department. After processing the department, the response process resets the value in element XXX-WK-FIRST-TIME from N to Y.

Process module XXXDUPD-ENTER is shown below. Statements that retrieve the specified department from the database, modify the department, and delete the department are organized into subroutines.

```
READY USAGE-MODE UPDATE.

- IF AGR-CURRENT-RESPONSE NE SPACES AND NO FIELDS CHANGED AND XXX-WK-FIRST-TIME EQ 'Y'
  THEN
    EXECUTE NEXT FUNCTION.

- IF XXX-WK-FIRST-TIME EQ 'Y'
  THEN
    CALL FRSTRTN.

- IF AGR-CURRENT-FUNCTION EQ 'MODDEP'
  THEN
    CALL MODRTN.

  ELSE
    CALL DELRTN.

DEFINE FRSTRTN.

OBTAIN CALC DEPARTMENT.

- IF DB-REC-NOT-FOUND
  THEN
    DISPLAY MSG TEXT
    'DEPARTMENT DOES NOT EXIST--SPECIFY A DIFFERENT DEPARTMENT'.

ELSE DO.

- MOVE 'N' TO XXX-WK-FIRST-TIME.
- PROTECT FIELD DEPT-ID-0410 TEMPORARY.

- IF AGR-CURRENT-FUNCTION EQ 'MODDEP'
  THEN
    DISPLAY MSG TEXT
    'MODIFY DEPARTMENT AND PRESS ENTER (PA2 TO CANCEL)'.

  ELSE
    DISPLAY MSG TEXT
    'PRESS ENTER TO DELETE THIS DEPARTMENT (PA2 TO CANCEL)'.

END.

DEFINE MODRTN.

MODIFY DEPARTMENT.

INITIALIZE (DEPARTMENT).

MOVE 'Y' TO XXX-WK-FIRST-TIME
DISPLAY MSG TEXT
  'DEPARTMENT MODIFIED--SPECIFY ANOTHER DEPARTMENT TO MODIFY'.
```
DEFINE DELRTN. -- 12
ERASE DEPARTMENT ALLOWING ('0230'). -- 13
MOVE 'Y' TO XXX-WK-FIRST-TIME. -- 14
IF ERROR-STATUS EQ '0230'
   THEN
      DISPLAY MSG TEXT
      'CANNOT DELETE THIS DEPARTMENT--SPECIFY ANOTHER DEPARTMENT'.
   ELSE DO.
      MOVE 'Y' TO XXX-WK-FIRST-TIME.
      INITIALIZE (DEPARTMENT).
      DISPLAY MSG TEXT
      'DEPARTMENT DELETED--SPECIFY ANOTHER DEPARTMENT TO DELETE'.
END.

1 Transfers control to another application function when the user enters a valid application response
and doesn't try to input other information on the screen.

2 The first time this process is executed for a given department. subroutine FRSTRTN is called.

3 The next time the process is executed for the department, the appropriate subroutine is called
(MODRTN for function MODDEP or DELRTN for function DELDEP).

FRSTRTN subroutine

4 Subroutine FRSTRTN begins.

5 Uses the department ID supplied by the user to retrieve the department.

6 If the specified department does not exist in the database, the dialog's map is redisplayed with the
DEPARTMENT DOES NOT EXIST error message.

7 If the department does exist, the flag in XXX-WK-FIRST-TIME is set to N (NO) and the map field that
displays department ID numbers is temporarily protected from user input.

8 The retrieved department record is displayed on the dialog's map with an appropriate message.

MODRTN subroutine

9 Subroutine MODRTN begins.

10 The department is modified in the database based on the user's input.

11 After the department is modified, dialog buffers for DEPARTMENT data are initialized, the flag in
XXX-WK-FIRST-TIME is reset to Y (YES), and the map is redisplayed with a confirming message.

DELRTN subroutine
Subroutine DELRTN begins.

The department is deleted from the database. Your code tests for errors that cause error code 0230 (see below).

Resets the flag in XXXWK-FIRST-TIME to Y (YES).

If status code 0230 is returned, the department cannot be deleted because it owns other records (for example, employee records). In this case, the map is redisplayed with a message and the department is not deleted.

After the department is deleted, dialog buffers are initialized and the dialog's map is redisplayed with a confirming message.

Specifying Basic Information for the Process Module

To specify basic information for process module XXXDUPD-ENTER, you use the IDD Process Entity screen:

```
-> proc
```

Enter the screen identifier (PROC) in the command area.

Enter the indicated specifications on the Process Entity screen. Don't forget to deselect the DISPLAY action.

```
IDD REL 15.0
*** PROCESS ENTITY ***
PROC
->
DICT=DEMO
DISPLAY PROCESS NAME.....: xxdupd-enter
MODIFY
ADD VERSION NUMBER....: 1 HIGHEST NEXT HIGHEST
DELETE LOWEST NEXT LOWEST
DESCRIPTION.....: retrieve dept and then mod/del dept
```

Entering Source Statements for the Process

To enter source commands for a process module, you display and use the Process Source screen, as shown below for process XXXDUPD-ENTER:

```
-> srcs
```

SRCE identifies the Process Source screen.

The Process Source screen is displayed. Enter a full screen of source statements. Enter keywords, periods, and quotes as shown below. Don't type characters beyond column 72.

For more information on entering, moving, and deleting text on the Process Source screen, refer to CA IDMS Common Facilities Section.
After you type the first page of source commands for the process module, **open up new lines at the end of the Process Source screen** by performing the following steps:

1. **Move the cursor** to the line containing the final source command on the screen.

2. **Press [PF4]** (default) to insert new lines after the cursor.

You can press [PF5] (default) at any time to apply the new lines to the work file maintained by the IDD menu facility.

For process module `XXXDUPD-ENTER`, you open up new lines at the end of the Process Source screen and enter more source commands. Place the cursor on the final line of text and press the control key that inserts new lines on IDD screens at your site.

```
--> IDD REL 15.0  *** PROCESS SOURCE ***  SRCE
    NO DATA LINES CURRENTLY EXIST
    PROCESS 'XXXDUPD-PREMAP' VERSION 1

-----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
ready usage-mode update
!
if agr-current-response ne spaces and no fields changed and xxx-wk-first-time eq 'y'
then
    execute next function.
!
if xxx-wk-first-time eq 'y'
then
    call frstrtn.
!
if agr-current-function eq 'moddep'
then
    call modrtn.
else
    call delrtn.
!
define frstrtn.

[PF4]
```

Add source statements to the end of the process. Notice that the final line from the previous page of the Process Source screen is displayed as the first line of this screen.

```
--> IDD REL 15.0  *** PROCESS SOURCE ***  SRCE
    INSERTING NEW DATA LINES
    PROCESS 'XXXDUPD-PREMAP' VERSION 1

-----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
DEFINE FRSTRTN.
obtain calc department.
if db-rec-not-found
then
display msg text
    'department does not exist--specify a different department'.
!
else do.
move 'n' to xxx-wk-first-time
protect field dept-id-0410 temporary.
!
if agr-current-function eq 'moddep'
then
display msg text
    'modify department and press enter (pa1 to cancel)'.
!
else
```
display msg text
   'press enter to delete this department (pa1 to cancel)'.

[PF4]

Place the cursor on the final source line and use [PF4] to insert another page of source statements.

Entering the Final Page of Source Statements

Type source statements on the screen. Don't forget to place the periods outside of the single quote for DISPLAY commands.

After you press [Enter], IDD adds the process module source commands to the data dictionary, and then redisplays the Process Source screen with a message like:

PROCESS 'XXXDUPD-ENTER' VERSION 1 MODIFIED

It is a good idea at this stage to inspect the process module for any syntax errors (such as omitted single quotation marks or periods). You can page the Process Source screen forward and backward:
• Press [PF8] (default) to page forward.

• Press [PF7] (default) to page backward.

If you notice any errors, you can type over the errors to correct them. After you correct all errors, you can press [Enter] again.

After you finish defining process module XXXDUPD-ENTER, you can define process XXXDUPD-PA2.

Step 3 Define Process Module XXXDUPD-PA2

Process module XXXDUPD-PA2 is the final process module that you will create for dialog XXXDUPD. Sample commands for process module XXXDUPD-PA2 are shown below. Statements in this process module allow the end user to cancel the current modification or deletion operation.

MOVE 'Y' TO XXX-WK-FIRST-TIME. INITIALIZDE (DEPARTMENT).

IF AGR-CURRENT-FUNCTION EQ 'MODDEP'
  THEN
    DISPLAY MSG TEXT 'MODIFICATION CANCELLED--SPECIFY A DEPARTMENT TO MODIFY'.
  ELSE
    DISPLAY MSG TEXT 'DELETION CANCELLED--SPECIFY A DEPARTMENT TO DELETE'.

1 These statements set the flag in XXX-WK-FIRST-TIME to Y and then initialize the dialog buffers for DEPARTMENT data.

2 For the MODDEP function, the dialog's map is redisplayed with this MODIFICATION CANCELLED message.

3 For the DELDEP function, the dialog's map is redisplayed with this DELETION CANCELLED message.

To define process module XXXDUPD-PA2, you use the Process Entity and Process Source screens:

-> proc

Enter the screen identifier (PROC) in the command area.

Enter the indicated specifications on the Process Entity screen. Don't forget to deselect the DISPLAY action.

- IDD REL 15.0 *** PROCESS ENTITY *** PROC
  ->
  DICIT=DEMO
  DISPLAY MODIY PROCESS NAME.....: xxxdupd-pa2
  X ADD VERSION NUMBER....: 1 HIGHEST NEXT HIGHEST
  _ DELETE _ LOWEST NEXT LOWEST
  DESCRIPTION.....: cancel mod/del department operation

[Enter]

After specifying basic information for process module XXXDUPD-PA2, you can proceed to add source commands on the Process Source screen:
SRCE identifies the Process Source screen.

The Process Source screen is displayed. Enter a full screen of source statements. Enter keywords, periods, and quotes as shown. Don't type characters beyond column 72.

```
-> srcs
```

After you press [Enter], IDD adds the process module source commands to the data dictionary and then redisplays the Process Source screen with a message like:

```
PROCESS 'XXXDUPD-PA2' VERSION 1 MODIFIED
```

If you notice any errors on the redisplayed screen, you can type over the errors to correct them, and then press [Enter] again.

After you finish defining process module XXXDUPD-PA2, you can exit from the IDD menu facility. In this sample session, you will transfer from IDD to ADSC in order to associate the process modules you defined above with dialog XXXDUPD. To do this, you use the SWITCH command:

```
-> switch adsct
```

Completing Dialog XXXDUPD Using ADSC

Now that you have defined process modules for dialog XXXDUPD, you can complete the dialog. To do this, you will perform the following steps:

1. Retrieve dialog XXXDUPD.
2. Add a subschema to the dialog.

3. Define dialog options for use during development.

4. Add a work record to the dialog.

5. Add premap and response processes to the dialog.

6. Recompile the dialog load module.

Each of these steps is presented below.

**Step 1 Retrieve Dialog XXXDUPD**

In order to retrieve a dialog, you must be using ADSC.

If you did not transfer to ADSC earlier in this section when you exited from the IDD menu facility, you need to invoke ADSC. To do this, enter the task code (for example, **ADSC**T) for ADSC. For more information on invoking ADSC, see [Step 1: Invoke ADSC (see page 243)](#).

ADSC begins by displaying the Main Menu screen. To retrieve a dialog, you identify the dialog on a blank **Main Menu** screen:

```
Add  Modify  Compile  Delete  Display  Switch
```

CA ADS Online Dialog Compiler

CA, Inc.

Dialog name ......... xxdupd
Dialog version ........ 1
Dictionary name ....... demo
Dictionary node ........ ________
Screen ............. 1  1. General options
                  2. Assign maps
                  3. Assign database
                  4. Assign records and tables
                  5. Assign process modules

Copyright (C) 2003 CA, Inc.

Command ===>  Enter F1=Help  F3=Exit  F10=Action

[PF10]

To check the application out for modification, position the cursor on the **Modify** item on the action bar and press [Enter]. You can position the cursor on **Modify** by:

- Pressing [PF10] to move to the action bar and then tabbing to **Modify** and pressing [Enter]

- Tabbing to **Modify** and pressing [Enter]

- Typing **modify** on the command line and pressing [Enter]

  Add  Modify  Compile  Delete  Display  Switch

._____________________________________________________________________________.
Press [Enter] to check the application out.

Note: If the dialog has been checked out to another developer and has not been released by that developer, you will not be able to check it out.

After you press [Enter], ADSC displays dialog XXXDUPD on the Main Menu screen, along with the following messages:

| NO SCHEMA/SUBSCHEMA - NO DATABASE CALLS ALLOWED |
| NO ERRORS DETECTED |

The NO ERRORS DETECTED message is the message you should look for after retrieving a dialog definition. After you successfully retrieve XXXDUPD, you can associate a subschema with the dialog.

Step 2 Add a Subschema

In this step, you will associate a subschema with dialog XXXDUPD so that process modules for the dialog can retrieve, modify, and delete departments in the database at run time.

You use the Database Specifications screen to associate a subschema with a dialog. For example:

Accessing the Database Specifications Screen

To access the Database Specifications screen, you enter 3 at the Screen prompt on the Main Menu.
Sample Database Specifications Screen

The Database Specifications screen is displayed.

```
Database Specifications
Dialog XXXDUPD Version 1
Subschema ............... empss01
Schema ..................
Version ................. 1
Access module ........... XXXDADD
```

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

For more information on subschemas, see Step 3: Add a Subschema (see page 303).

When you press the [Enter] key, ADSC associates the named subschema with the dialog if there are no errors, and then redisplays the screen with the following message:

N0 ERRORS DETECTED

If a different message is displayed, read the message to determine the problem. If you did not specify a schema name, ask others at your site whether a schema name is required. After you type new values, press [Enter] again.

After you successfully associate a subschema with the dialog, you can proceed to add work record XXX-WK-RECORD to the dialog.

Step 3 Define Dialog Options

You use the Options and Directives screen to specify options to help you develop and debug dialogs. In this section, you use the Options and Directives screen to add a symbol table and a diagnostic table to dialog XXXDUPD. These tables are often useful when debugging dialogs.
You can display and use the Options and Directives screen as shown: The diagnostic table is enabled by default. Enable the symbol table.

Options and Directives

Message prefix . . . . . . . . . . . . DC
Autostatus record . . . . . . . . . . . . ADSO-STAT-DEF-REC
Version . . . . . . . . . . . . . . . . . . 1
Options and directives . . . . . Mainline dialog
/ Symbol table is enabled
/ Diagnostic table is enabled
/ Entry point is premap
/ COBOL moves are enabled
/ Activity logging
/ Retrieval locks are kept
/ Autostatus is enabled

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

After you press [Enter], the Options and Directives screen displays the following message to inform you that your specifications contain no errors:

INPUT HAS BEEN SUCCESSFULLY PROCESSED

For more information on the Options and Directives screen, see "Step 2: Specify Dialog Options".

Step 4 Add a Work Record

Process modules that you defined for dialog XXXDUPD use element XXX-WK-FIRST-TIME. To enable the process modules to use the element at run time, you add to the dialog the work record (XXX-WK-RECORD) that contains the element.

Some work records define data to be displayed on the dialog's map. In this case, the work record is automatically added to the dialog when the map is added.

Work record XXX-WK-RECORD does not define data for display on the dialog's map. Therefore, you need to explicitly add the record to the dialog. To do this, you use the ADSC Records and Tables screen.

You access and use the Records and Tables screen as shown:

Records and Tables

Name xxx-wk-record Version Work New copy Drop

---------------------- ----- --- --- ---
---------------------- ----- --- --- ---
---------------------- ----- --- --- ---
When you press [Enter], ADSC associates the named work record with the dialog if there are no errors, and then redisplays the Records and Tables screen with a message:

- **Records and Tables PROCESSED SUCCESSFULLY** is displayed when you successfully associate a work record with the dialog.

- **A different message** is displayed if ADSC detects any errors. In this case, read the message to determine the problem. If the work record was provided to you by other developers at your site, find out if you need to supply a version number for the record under the **Version** prompt. You can type over any mistakes and press [Enter] again.

After you successfully associate work record XXX-WK-RECORD with dialog XXXDUPD, you can proceed to add process modules to the dialog.

### Step 5 Add Premap and Response Processes

You use the ADSC **Process Modules** screen to associate premap and response processes with a dialog.

Earlier in this section, you defined a premap process, XXXDUPD-PREMAP, and two response processes for dialog XXXDUPD: XXXDUPD-ENTER and XXXDUPD-PA2.

The premap process will execute before the map is displayed to the end user.

When the end user presses a control key to input data on the dialog’s map, one and only one of the response processes can be executed. To make it easy for the end user to select the appropriate response process, you will associate each response process with its own control key.

As an application developer, you associate control keys with response processes when you add the processes to the dialog:

- When you add **XXXDUPD-ENTER**, you will specify that this response process is invoked when the end user presses the [Enter] key.

- When you add **XXXDUPD-PA2**, you will specify that this response process is invoked when the end user presses [PA2]:

  You access the Process Modules screen by entering **5** at the **Screen** prompt on the Main Menu and pressing [Enter].

**Process Modules Screen**

The Process Modules screen is displayed. You type the name of the process module after the **Name** prompt:
When you press [Enter], ADSC compiles the process module source code and adds the compiled representation of each module to the dialog if there are no errors. ADSC redisplays the Process Modules screen with a message:

- **PROCESS HAS COMPILED SUCCESSFULLY** is displayed when the compiled process module was successfully added to the dialog as a premap process.

- **TO SEE ERRORS SELECT DISPLAY OR PRINT** is displayed when ADSC cannot successfully compile the process module because of errors.

In this case, you display and correct errors in the process module, as discussed in Correct Errors in Process Modules (see page 311).

After the process modules have compiled successfully, you can recompile the dialog.

### Step 6 Recompile the Dialog

In this section, you completed dialog XXXDUPD by adding a subschema, dialog options, a work record, a premap process, and two response processes.

To update the load module for dialog XXXDUPD, recompile the dialog by selecting the **Compile** activity:

Add Modify Compile Delete Display Switch

1. Compile log Compiler
2. Display messages

international, Inc.
F3=Exit

<table>
<thead>
<tr>
<th>Dialog name . . . . . . .</th>
<th>XXXDUPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog version . . . . .</td>
<td>1</td>
</tr>
<tr>
<td>Dictionary name . . . . .</td>
<td>DEMO</td>
</tr>
<tr>
<td>Dictionary node . . . . .</td>
<td>_______</td>
</tr>
<tr>
<td>Screen . . . . . . . . . .</td>
<td>1  1. General options</td>
</tr>
</tbody>
</table>
2. Assign maps
3. Assign database
4. Assign records and tables
5. Assign process modules

Command ===> 
Enter F1=Help F3=Exit F10=Action

When you press [Enter], ADSC attempts to recompile the dialog. ADSC then displays a message:

- **DIALOG HAS BEEN MODIFIED** is displayed on the Dialog Definition screen when the dialog was successfully recompiled.

- **A different message** is displayed if an error in the dialog definition prevents ADSC from recompiling the dialog.
  
  In this case, correct the errors and then recompile the dialog as described above.

After you successfully recompile dialog XXXDUPD, you can exit from ADSC by using [PF3].

After you exit from ADSC, you can execute the application.

**Test the Department Application**

Once you enhance dialog XXXDUPD, the sample Department application is complete. You can execute the application to test out all application features and see how end users would use them, as follows:

- **Invoke the Department Application** (see page 361)
- **Display MODDEP Function** (see page 362)
- **Verify the Modifications** (see page 363)
- **Display DELDEP Function** (see page 366)

When testing functions MODDEP and DELDEP, you can modify and delete the sample departments that you added when you tested function ADDDEP in section "Adding Process Logic to a Dialog". When you modify departments in this section, it is a good idea to **write down** the changes that you make.

**Invoke the Department Application**

To invoke the Department application from IDMS-DC/UCF, you enter the task code (XXXDEPT) for the application. For example, when using IDMS-DC, you invoke the Department application as shown:

ENTER NEXT TASK CODE: xxxdept

For more information on invoking the application, see **Test the Application (see page 250)**.
Display MODDEP Function

You can display the MODDEP function and modify a department, as shown below for a sample department named TEST DEPARTMENT:

x MOD

[Enter]

Select the MOD response to display MODDEP.

The MODDEP function screen is displayed. Specify the department to be modified.

FUNCTION: MODDEP
DATE....: 10/30/99
DEPARTMENT INFORMATION

DEPARTMENT ID .......: 9876
NAME ......: TEST DEPARTMENT -- purchasing
HEAD ID ..: 0000

RESPONSE:
MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT
[Enter]

You defined the message in premap process XXXDUPD-PREMAP.

Modify the displayed information by typing over and erasing old values.

Remember that to promote data integrity, you protected the DEPARTMENT ID variable field (which contains a department's unique ID number) from user updates.

FUNCTION: MODDEP
DATE....: 10/30/99
DEPARTMENT INFORMATION

DEPARTMENT ID .......: 9876
NAME ......: TEST DEPARTMENT -- purchasing
HEAD ID ..: 5555

RESPONSE:
MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT
[Enter]

The department is modified.

Notice the message you defined in response process XXXDUPD-ENTER for display when department data has been modified.
The process logic that you defined for dialog XXXDUPD allows you to modify department records in two steps:

1. The first time you press [Enter], response process XXXDUPD-ENTER retrieves the specified department from the database.

2. The second time you press [Enter], the same response process modifies information for the department in the database.

The following diagram shows how components for dialog XXXDUPD are executed at run time when you use the dialog to modify a department record. Response process XXXDUPD-ENTER executes twice when you use dialog XXXDUPD to modify a department.

Verify the Modifications

After you modify a department, you can verify that your modifications have been updated to the database by specifying the department ID again:

DEPARTMENT ID ........: 9876

[Enter]

Modified values are retrieved from the database and the modified department is redisplayed.
DEPARTMENT ID .......: 9876  
NAME ......: TEST DEPARTMENT -- PURCHASING  
HEAD ID ..: 5555

RESPONSE:

MODIFY DEPARTMENT AND PRESS ENTER (PA2 TO CANCEL)

Instead of modifying this department again, you can try pressing [PA2] to see how process module XXXDUPD-PA2 allows you to cancel a modification operation:

FUNCTION: MODDEP
DATE....: 10/30/99

DEPARTMENT INFORMATION

DEPARTMENT ID .......: 0000  
NAME ......:  
HEAD ID ..: 0000

RESPONSE:

MODIFICATION CANCELLED--SPECIFY A DEPARTMENT TO MODIFY

You defined this message in response process XXXDUPD-PA2 to confirm cancellation of a modification operation.

When you press [PA2], response process XXXDUPD-PA2 is executed. This response process cancels your current operation without modifying the database. The following diagram shows how components for dialog XXXDUPD are executed when you press [PA2] to cancel a modification operation. XXXDUPD-PA2 and cancel a modification operation.
IDMSDB--Executing the Application (3)

While executing the Department application, you also can test out how dialog XXXDUPD handles deletion operations. To do this:

1. Transfer from MODDEP to DELDEP.
2. Then, specify the ID number of a department to be deleted.

Deleting a Sample Department

Function DELDEP allows you to delete existing departments from the database. You should be careful, when you test this function, that you delete only your own sample departments from the database.

You can use the DELDEP function to delete a department.

Specify the department to be deleted by entering the ID number for an existing department.

DEPARTMENT ID........: 9876 [Enter]

The department record is displayed for your verification.

FUNCTION: DELDEP
DATE.....: 10/30/99

DEPARTMENT INFORMATION

DEPARTMENT ID ........: 9876
NAME ......: TEST DEPARTMENT -- PURCHASING
HEAD ID ..: 5555

RESPONSE:
PRESS ENTER TO DELETE THIS DEPARTMENT (PA2 TO CANCEL)
[Enter]

When you press [Enter], the screen is redisplayed with the message you defined in response process XXXDUPD-ENTER:

DEPARTMENT DELETED--SPECIFY ANOTHER DEPARTMENT TO DELETE

The process logic that you defined for dialog XXXDUPD allows you to delete a department record in two steps:

1. The first time you press [Enter], process module XXXDUPD-ENTER retrieves and displays the specified department.
2. The second time you press [Enter], process module XXXDUPD-ENTER deletes the department from the database.

You can verify that process module XXXDUPD-ENTER actually deleted the department from the database by specifying the department ID number again:

DEPARTMENT ID........: 9876

[Enter]

When you press [Enter], the screen is redisplayed with the message you defined in response process XXXDUPD-ENTER:

DEPARTMENT DOES NOT EXIST--PRESS ENTER TO SPECIFY ANOTHER

While testing dialog XXXDUPD, you also can verify that [PA2] allows an end user to cancel a deletion operation. To do this, you specify an existing department, as shown:

DEPARTMENT ID........: 9876

[Enter]

Then, when dialog XXXDUPD displays the department record, you press [PA2].

Display DELDEP Function

Before you exit from the application, you can use the DELDEP function to delete all sample departments that you’ve added to the database.

While you are executing this production version of the Department application, you also can test out all other capabilities of the sample Department application to see how the application allows you to perform all operations needed to add, modify, and delete departments in the database.

When you are ready to exit from the Department application, you can select the EXIT response.

Summary 8

The sample Department application is now fully developed. The steps you performed while developing the Department application throughout this section could have been performed in a different order, depending on the preferences and requirements of the site. For example, you could have added process logic to dialogs before associating the dialogs with ADSA application functions.
In Part II of this section, "Developing the Prototype," you began developing the Department application based on a structure diagram for the application. You created a prototype of the application by defining components for the application:

1. **You defined the executable application structure** in Section 8, "Defining an Application Structure Using ADSA". The structure consists of:
   - **Functions** (for example, MODDEP) that represent units of work (such as dialogs) to be performed by the application
   - **Responses** (for example, EXIT) that define run-time paths between functions in the application
   - **A task code** (XXXDEPT) that defines an entry point for the application

2. **You defined a map** in Section 9, "Defining a Screen Display Using MAPC". The map allows dialogs in the application to display department information.

3. **You defined skeleton versions of dialogs** in Section 10, "Defining Dialogs Using ADSC". At run time, these dialogs are executed when control passes to the application functions that invoke them.

Based on tests of the completed prototype, you modified the prototype to make the application easier to use. You changed flow of control and screen displays without having to alter any code.

After the prototype application was approved, you developed a fully functional application from the prototype in Part III of this section, "Enhancing the Application Definition." You added process logic and a work record to the application’s skeleton dialogs:

1. **You defined process modules for dialog XXXDADD** in Section 13, "Adding Process Logic to a Dialog" and Section 14, "Modifying Process Logic in a Dialog". These processes allow end users to add new departments to the database.

2. **You defined a work record** in Section 14, "Modifying Process Logic in a Dialog". The record stores a value that determines the processing performed by dialog XXXDUPD at run time.

3. **You defined process modules for dialog XXXDUPD** in this section. These processes allow end users to modify and delete departments in the database.

After completing the Department application, you executed the completed application to test its full capabilities.

Even though you have tested the Department application, you still may need to make modifications either now or in the future. For example, consider the following sources of change:

- **Tests and preliminary use of the application** can cause developers, end users, and managers to suggest changes.
  For example, end users of the Department application might prefer blanks to be displayed instead of zeros (0000) in fields where they enter numeric values (for example, the DEPARTMENT ID variable field).
  End users might request that execution of the application be changed from STEP to FAST mode. In FAST mode, the end user can enter values on one screen and transfer to another screen at the
same time.
End users might have difficulty remembering the ID numbers for all departments to be modified or deleted. One solution would be to add a dialog function that alphabetically lists all departments and allows the end user to select a department from the list.

- Changes in regulations, updated business functions, and other planned and unplanned developments can require developers to modify an application.
  For example, new contractors might require the collection of additional information for departments.
  Long-term plans for the sample Department application could call for you to add EMPLOYEE and OFFICE functions to the existing application structure.

CA ADS application development tools can be used at any time to modify individual application components and to add new components to the application. For example, you can add EMPLOYEE and OFFICE functions to the Department application by defining maps and process modules for EMPLOYEE and OFFICE dialogs. When the dialogs are developed, you can incorporate the dialogs into the application structure by using ADSA to add the necessary functions and responses to the application.

Sample Application Components

Components in the Sample Application

This section provides information for all components created for and used by the sample Department application. Components are presented in the order in which you first define them or use them in this section. Names of components in this table that begin with XXX can be changed when you define the components. For example, you can use your initials instead of XXX.

<table>
<thead>
<tr>
<th>Component type</th>
<th>Sample name</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application structure</td>
<td>XXXAPP</td>
<td>Defined in Steps 2 through 8 of Section7. Modified in Steps 2 through 4 of Section10. Contains: BACK response, EXIT response, ADD response, MOD response, DEL response, ADDDEP function, MODDEP function, DELDEP function, DEPTMENU function, XXXDEPT task code</td>
</tr>
<tr>
<td>Application responses</td>
<td>BACK</td>
<td>Defined in Steps 4 and 5 of Section7: Assigned key: CLEAR, Function invoked: POP, Response type: GLOBAL</td>
</tr>
<tr>
<td>Application responses</td>
<td>EXIT</td>
<td></td>
</tr>
</tbody>
</table>

Sample Application Components
<table>
<thead>
<tr>
<th>Component type</th>
<th>Sample name</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Defined in Steps 4 and 5 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigned key: [PF9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function invoked: QUIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response type: GLOBAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified in Steps 2 and 3 of Section 10:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigned key: [PF3]</td>
</tr>
<tr>
<td>Application responses</td>
<td>ADD</td>
<td>Defined in Steps 4 and 5 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigned key: [PF4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function invoked: ADDDEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response type: LOCAL</td>
</tr>
<tr>
<td>Application responses</td>
<td>MOD</td>
<td>Defined in Steps 4 and 5 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigned key: [PF5]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function invoked: MODDEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response type: LOCAL</td>
</tr>
<tr>
<td>Application responses</td>
<td>DEL</td>
<td>Defined in Steps 4 and 5 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigned key: [PF6]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function invoked: DELDEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response type: LOCAL</td>
</tr>
<tr>
<td>Application functions</td>
<td>ADDDEP</td>
<td>Defined in Steps 4 and 6 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dialog name: XXXDADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function type: 1 (dialog)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid responses: BACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified in Steps 2 and 4 of Section 10:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid responses: MOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT</td>
</tr>
<tr>
<td>Application functions</td>
<td>MODDEP</td>
<td>Defined in Steps 4 and 6 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dialog name: XXXDUPD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function type: 1 (dialog)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid responses: BACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT</td>
</tr>
<tr>
<td>Application functions</td>
<td>DELDEP</td>
<td>Defined in Steps 4 and 6 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dialog name: XXXDUPD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function type: 1 (dialog)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid responses: BACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT</td>
</tr>
<tr>
<td>Application functions</td>
<td>DEPTMEN</td>
<td>Defined in Steps 4 and 6 of Section 7:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Function type: 3 (menu)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid responses: ADD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT</td>
</tr>
<tr>
<td>Task code</td>
<td>XXXDEPT</td>
<td>Defined in Step 7 of Section 7:</td>
</tr>
<tr>
<td>Database record</td>
<td>DEPART</td>
<td>Assoc. function: DEPTMENU</td>
</tr>
</tbody>
</table>

<p>| DEPART | Added to map XXXMAP in Step 3 of Section 8; describes department information to be stored in the database. |</p>
<table>
<thead>
<tr>
<th>Component type</th>
<th>Sample name</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>XXXMAP</td>
<td>Defined in Steps 1 through 9 of Section 8. Associated records: DEPARTMENT ADSO-APPLICATION-GLOBAL-RECORD Modified in Section 11.</td>
</tr>
<tr>
<td>Subschema</td>
<td>EMPSS01</td>
<td>Added to dialog XXXDADD in Step 1 in Section 12. Added to dialog XXXDUPD in &quot;Completing dialog XXXDUPD using ADSC&quot; in Section 15; contains the DEPARTMENT database record and makes it available to dialogs using navigational DML statements.</td>
</tr>
<tr>
<td>Dialogs</td>
<td>XXXDADD</td>
<td>Defined in Steps 2 and 3 of Section 9: Associated map....: XXXMAP Updated due to modifications in map XXXMAP in Steps 1 through 3 in Section 11. Enhanced in &quot;Adding process modules to dialogs using ADSC&quot; in Section 12: Subschema.........: EMPSS01 Premap process....: XXXDADD-PREMAP Response process.: XXXDADD-RESPONSE Modified in &quot;Updating modified process modules in dialogs using ADSC&quot; in Section 13.</td>
</tr>
<tr>
<td>Dialogs</td>
<td>XXXDU</td>
<td>Defined in Step 5 of Section 9: Associated map....: XXXMAP Updated due to modifications in map XXXMAP in &quot;Updating modified maps in dialogs using ADSC&quot; in Section 11. Enhanced in &quot;Completing dialog XXXDUPD using ADSC&quot; in Section 15: Subschema.........: EMPSS01 Premap process....: XXXDUPD-PREMAP Response processes: XXXDUPD-ENTER XXXDUPD-PA2</td>
</tr>
<tr>
<td>Process module</td>
<td>XXXDA</td>
<td>Defined in &quot;Defining process modules using IDD&quot; in Section 12: Purpose.: Premap process for dialog XXXDADD</td>
</tr>
<tr>
<td>Process module</td>
<td>XXXDA</td>
<td>Defined in &quot;Defining process modules using IDD&quot; in Section 12: Purpose.: Response process for dialog XXXDADD</td>
</tr>
<tr>
<td>Element</td>
<td>XXX-WK-FRST</td>
<td>Defined in Step 2 of Section 14. Purpose.: Stores a single-character status value (Y or N) for process modules in dialog XXXDUPD Usage ...: DISPLAY Picture.: X Added to work record XXX-WK-RECORD in Step 3 of Section 14.</td>
</tr>
</tbody>
</table>
Component type | Sample name | Characteristics
--- | --- | ---
Work record | XXX-WK-RECORD | Defined in Step 3 of Section 14:
Purpose...: Work record for process modules
Element...: XXX-WK-FIRST-TIME

Process modules | XXX-DU-PD-PREMAP | Defined in "Defining process modules using IDD" in Section 15:
Purpose...: Premap process for dialog XXXDUPD

Process modules | XXX-DU-PD-ENTER | Defined in "Defining process modules using IDD" in Section 15:
Purpose...: Response process for dialog XXXDUPD

Process modules | XXX-DU-PD-PA2 | Defined in "Defining process modules using IDD" in Section 15:
Purpose...: Response process for dialog XXXDUPD

---

Development Tools in the CA ADS Environment

You use the following development tools when you define a CA ADS application:

- **CA ADS application compiler (ADSA)** -- Used to define the executable structure of an application
- **CA ADS dialog compiler (ADSC)** -- Used to define the dialogs that display and request data at runtime
- **Online mapping facility (MAPC)** -- Used to define the screens (maps) displayed by dialogs at runtime
- **Integrated Data Dictionary (IDD) menu facility** -- Used to define records and process modules

**Types of Definitions**

The types of definitions created by using ADSA, ADSC, MAPC, and IDD are listed in the following table. Following a general discussion of the use of CA ADS development tools, operations that are commonly performed when using ADSA, ADSC, MAPC, and IDD are presented for each tool. The definitions created by each CA ADS development tool are listed in this table.

<table>
<thead>
<tr>
<th>Development tool</th>
<th>Definitions created</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA ADS application compiler (ADSA)</td>
<td>Application structures composed of: Functions Responses Task codes</td>
</tr>
<tr>
<td>CA ADS dialog compiler (ADSC)</td>
<td>Dialogs</td>
</tr>
<tr>
<td>Online mapping facility (MAPC)</td>
<td>Maps</td>
</tr>
</tbody>
</table>
CA ADS Development Tools

### Contents
- Invoking Development Tools (see page 373)
- Exiting from Development Tools (see page 374)

ADSA, ADSC, MAPC, and the IDD menu facility all display definition screens to help you define application components. Prompts and default values are displayed whenever possible. When using definition screens, you can:

- **Type specifications** in fields on the screen.
  Specification fields typically precede or immediately follow prompts. For example, when naming an application on the ADSA Main Menu screen:
  
  Application name . . . . xxxappl

  You type the application name in the specification field that follows the Application name prompt.

  You enter these specifications when you press [Enter] or any valid PF key (such as [PF5]).

- **Move the cursor from prompt to prompt** by using the forward tab key, backward tab key, or return key. Additionally, you can use any of the cursor movement keys to move the cursor up, down, left, or right.

- **Move from screen to screen** by selecting an activity from the Main Menu or pressing a control key:
  
  - **You can select an activity** from the Main Menu.
    For example, on the ADSA Main Menu screen, you can select the Task Codes screen by entering 4 opposite the Screen prompt.

  - **You can press a control key** that is defined for use in the current development tool.
    For example, [PF5] is used to display the next screen in the definition process.

---

1 At some sites, the application developer is not responsible for creating data definitions.
2 For information on code and edit tables, refer to CA IDMS Mapping Facility Section.
You can invoke a CA ADS development tool from CA IDMS/DC or DC/UCF or from another CA ADS development tool. You can exit from a development tool at any time. Invoking and exiting from development tools are discussed below.

Invoking Development Tools

To invoke a CA ADS development tool, you must be signed on to DC/UCF. The method you use to invoke a development tool depends on your current location:

- **From DC/UCF**, you use the task code defined for the development tool.
  For example, if ADSAT is the task code for the CA ADS application compiler, you **invoke ADSA from CA IDMS/DC**:
  
  `ENTER NEXT TASK CODE:`
  `adsat [Enter]`

  Using the task code ADSAT means that you are invoking ADSA under TCF (the transfer control facility). Once you are in the ADSA tool under TCF, you can switch to another application development tool without returning to DC/UCF.

  **Note:** Using the task code **ADSA**, rather than ADSAT, also invokes ADSA. However, TCF is not involved and, because ADSA is not running under TCF, you are not able to switch to another application development tool without returning to DC/UCF first.

  For more information on task codes, see Application Development Tools (see page 180).

- **From another development tool** under the transfer control facility (TCF), use the command or function key that transfers control from that tool to another. The method used depends on the tool you currently are using:

  - **From ADSA, ADSC, or MAPC**, you can transfer directly to another tool by specifying the task code for the tool after the **Switch** activity on the action bar at the top of the Main Menu.

  - **From IDD**, you can transfer directly to another tool by typing the SWITCH command in the command area at the top of the screen, followed by the task code for that tool.
    For example, if MAPCT is the task code for the online mapping facility (MAPC), you can **transfer from IDD to MAPC**:
    `switch mapct [Enter]`

  - **From the TCF Selection Screen**, select the development tool's task code from the screen and press [Enter].
    For example, if ADSCT is the task code for the CA ADS dialog compiler, you can **invoke ADSC from the Selection Screen**:
    
    | CA TRANSFER CONTROL FACILITY | CAGJF0 |
    |-------------------------------|--------|
    | TRANSFER CONTROL FACILITY    |        |
    | Suspend TCF Session (PF9)    |        |
    | Terminate TCF Session (PF3)  |        |
    | *TCF TaskCodes*              |        |
    | Select one to start a new session |     |
    | DBNAME:                      | DEMO   |
    | DNNODE:                      |       |
    | *Suspended Sessions*         |        |
    | Select one to resume an old session |     |
    | TaskCode | Descriptor |
Exiting from Development Tools

You can exit from ADSA, ADSC, MAPC, or the IDD menu facility at any time during a definition session. Exit methods available for these development tools include those listed below:

- **The SWITCH command** (under TCF only) allows you to save the definition currently in progress when you exit from a development tool. You can use SWITCH to:
  - **Transfer** to another development tool as described above in Invoking Development Tools (see page 373).
  - **Exit** directly to DC/UCF:
    - **From ADSA, ADSC, or MAPC**, specify the SUSPEND keyword after the SWITCH TASK activity at the bottom of the screen:
      \[ \text{SWITCH TASK: suspend [Enter]} \]
    - **From IDD**, type the SUSPEND keyword after the SWITCH command in the command area at the top of the screen:
      \[ \text{switch suspend [Enter]} \]
      For more information on using the SWITCH command, refer to *CA IDMS Common Facilities Section*.

- **The [PF3] key** takes you backward through definition screens until you exit from the tool.

In **IDD**, the SUSPEND and QUIT commands allow you to exit. You type SUSPEND in the screen’s command area. Neither SWITCH nor QUIT allows you to specify a task code (for example, IDDMT) or any keywords (for example, the SUSPEND keyword).

Using ADSA

ADSA is an application design and prototyping tool used to define the structure of an application. The first screen in an ADSA session is the Main Menu screen.

**ADSA Main Menu Screen**

\[ \text{Add} \ \text{Modify} \ \text{Compile} \ \text{Delete} \ \text{Display} \ \text{Switch} \]
Enter information on the dialog after prompts in the Specification area. **To get from one ADSA screen to another**, you can either select the activity from the Screen Specification area on the Main Menu or, from other screens, press [PF5] to proceed through the definition.

Select an action by tabbing to the action bar or selecting with the command line.

The following table describes how to use ADSA to perform the following procedures:

- Adding an application
- Modifying an application
- Deleting an application
- Adding a response
- Modifying a response
- Deleting a response
- Adding a function
- Modifying a function
- Deleting a function
- Adding a task code
- Modifying a task code
- Deleting a task code

For more information on these and other ADSA procedures, refer to *CA ADS Reference Section*.

Instructions in the table assume that you have already invoked ADSA, as discussed earlier in this section.
### Adding an application

**Procedure**

Enter basic information about the application on a Main Menu screen:
- An application name
- A dictionary name/node (where applicable)

**Select the Add activity from the action bar** to register the application in the dictionary and check it out (reserve it) to the programmer.

**Add required components** on appropriate ADSA screens:
- Add responses and functions by using the Response/Function List screen as described under "Adding Responses and Functions" later in this table.

Further define responses by using the Response Definition screen, as described in "Adding a Response" later in this table.

Add functions by using the appropriate Function Definition screen, as described in "Adding a Function" later in this table.

Add task codes by using the Task Codes screen, as described in "Adding a Task Code" later in this table.

Optionally make additional specifications on other ADSA screens.

For more information on available ADSA screens, see the CA ADS Reference Section.

**Create a load module for the application** by selecting the Compile activity from the action bar on the Main Menu.

For an example of using ADSA to add an application, see Section, "Defining an Application Structure Using ADSA".

### Modifying an application

**Procedure**

Display the application definition (if not already displayed) by entering the following information on the application Main Menu screen:
- The application’s name
- A dictionary name/node (when applicable)

If the application has been released (after having been added), check the application out through the Check out option under the Modify activity on the action bar.

Modify application specifications, as necessary, on appropriate ADSA screens:
- Add and select responses and functions by using the Response/Function List screen, as described later in this table.
- Add, modify, or delete responses by using the Response Definition screen, as described later in this table.
- Add, modify, or delete functions by using the Function Definition screens, as described later in this table.
- Add, modify, or delete task codes by using the Task Codes screen, as described later in this table.

Recompile the application load module by selecting the Compile activity from the action bar on the Main Menu.

For an example of using ADSA to modify an application, see Modifying the Application Structure Using ADSA (see page 257).

### Deleting an application

**Procedure**

Enter the following information on the application Main Menuscreen:
- The application name
### Operation

<table>
<thead>
<tr>
<th>Functions, and task codes). To delete dialogs, maps, and process modules, see descriptions of ADSC, MAPC, and IDD later in this section.</th>
<th>A dictionary name/node (when applicable) <strong>Choose the Delete application option</strong> from the <strong>Delete</strong> activity on the action bar on the Main Menu. <strong>After you press [Enter]</strong>, ADSA displays a confirmation window so that the request to delete the application can be confirmed or rescinded. Confirm or reject the deletion. For an example of using ADSA, see Section8, &quot;Defining an Application Structure Using ADSA&quot;.</th>
</tr>
</thead>
</table>

### Adding response and function relationships

**Procedure**

Display the **Response/Function List screen** by entering a 2 opposite the **Screen** prompt on the Main Menu. **Enter the following information** on the **Response/Function List screen**:  
- The response name  
- An associated activity key (optional)  
- The name of the function invoked by the response3  
- The type of function (declaration, premap, response, default response)  
- The program or dialog name  
- A nonblank character to indicate which responses and functions need further definition  
For an example of using ADSA to add responses and functions, see Steps 4 through 6 in Section8, "Defining an Application Structure Using ADSA" of this section.

### Enhancing the response definition

**Procedure**

Select responses for further definition by entering a nonblank character next to the responses on the **Response/Function List screen**.  
**Display the Response Definition screen** by pressing [PF5] from the **Response/Function List screen**. **Enter the following information** on the **Response Definition screen**:  
- The response type (local or global)  
- The control command used to invoke the associated function  
For an example of using ADSA to further define responses, see Steps 4 and 5 in Section8, "Defining an Application Structure Using ADSA" of this section.

### Modifying a response

**Procedure**

Display the **responses and functions** on the **Response/Function List screen**.  
**Make any changes required**.  
**Optionally select the response to be modified** by placing a nonblank character next to the response and pressing [PF5] to access the **Response Definition screen**.  
**Modify any specifications** on the **Response Definition screen**, including:  
- The response type (global or local)  
- The control command used to invoke the function  
For an example of using ADSA to modify a response, see Steps 2 and 3 in Section, "Modifying the Application Structure Using ADSA".
<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleting a response</td>
<td>Display the responses and functions on the Response/Function List screen.</td>
</tr>
<tr>
<td></td>
<td>Select the response to be deleted by placing a nonblank character next to the response and pressing [PF5] to access the Response Definition screen.</td>
</tr>
<tr>
<td></td>
<td>Display the response definition (if not already displayed) on the Response Definition screen as described above in &quot;Modifying a response&quot; earlier in this table.</td>
</tr>
<tr>
<td></td>
<td>Place a nonblank character next to the Drop prompt</td>
</tr>
<tr>
<td></td>
<td>For an example of using the Response Definition screen, see &quot;Step 5: Further define application responses&quot; in &quot;Defining an Application Structure Using ADSA&quot;.</td>
</tr>
<tr>
<td>Enhancing the dialog function definition</td>
<td>Display the responses and functions on the Response/Function List screen.</td>
</tr>
<tr>
<td></td>
<td>Select the function to be defined by placing a nonblank character next to the function and pressing [PF5].</td>
</tr>
<tr>
<td></td>
<td>Display the Function Definition (Dialog) screen by pressing [PF5] from the Response/Function List screen.</td>
</tr>
<tr>
<td></td>
<td>Enter the following information on the Function Definition screen:</td>
</tr>
<tr>
<td></td>
<td>A description of the function</td>
</tr>
<tr>
<td></td>
<td>Valid responses for this function.</td>
</tr>
<tr>
<td></td>
<td>For an example of using ADSA to further define functions see &quot;Step 6: Further define application functions &quot;Defining an Application Structure Using ADSA&quot; of this section.</td>
</tr>
<tr>
<td>Enhancing the menu function definition</td>
<td>Display the responses and functions on the Response/Function List screen.</td>
</tr>
<tr>
<td></td>
<td>Select the menu function to be defined by placing a nonblank character next to the function and pressing [PF5].</td>
</tr>
<tr>
<td></td>
<td>Display the Function Definition (Menu) screen by pressing [PF5] from the Response/Function List screen.</td>
</tr>
<tr>
<td></td>
<td>Enter the following information on the Function Definition screen:</td>
</tr>
<tr>
<td></td>
<td>A description of the function</td>
</tr>
<tr>
<td></td>
<td>Specify heading text to be displayed on the runtime menu:</td>
</tr>
<tr>
<td></td>
<td>Enter the number of heading lines after the Heading lines prompt.</td>
</tr>
<tr>
<td></td>
<td>Enter the heading text in lines below the Heading line text prompt.</td>
</tr>
<tr>
<td></td>
<td>Identify the valid responses for this function.</td>
</tr>
<tr>
<td></td>
<td>Display the second page of the Function Definition (Menu) screen by pressing [PF8] from page 1.</td>
</tr>
<tr>
<td></td>
<td>Enter a nonblank character next to each response that is valid for the menu.</td>
</tr>
<tr>
<td></td>
<td>Customize the menu display for a menu function:</td>
</tr>
<tr>
<td></td>
<td>Optionally change the way in which responses are displayed on the runtime menu:</td>
</tr>
<tr>
<td></td>
<td>Optionally change the sequence of responses by typing new sequence numbers for the responses to be moved.</td>
</tr>
<tr>
<td></td>
<td>Optionally suppress the display of a valid response by entering zeros over the sequence number for the response; the response</td>
</tr>
<tr>
<td>Operation</td>
<td>Procedure</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>remains available from the menu even though it is not displayed.</td>
</tr>
<tr>
<td></td>
<td>For an example of changing the response sequence for a menu function, see &quot;Menu functions&quot; in &quot;Defining an Application Structure Using ADSA&quot;.</td>
</tr>
<tr>
<td></td>
<td>For an example of using ADSA to further define functions see &quot;Step 6: Further define application functions in &quot;Defining an Application Structure Using ADSA&quot; of this section.</td>
</tr>
<tr>
<td>Modifying a function</td>
<td>Display the responses and functions on the Response/Function List screen.</td>
</tr>
<tr>
<td></td>
<td>Make any changes required.</td>
</tr>
<tr>
<td></td>
<td>Optionally select the function to be further modified by placing a nonblank character next to the function and pressing [PF5] to</td>
</tr>
<tr>
<td></td>
<td>access the Function Definition screen.</td>
</tr>
<tr>
<td></td>
<td>Modify any specifications on the Function Definition screen, including:</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Associated dialog</td>
</tr>
<tr>
<td></td>
<td>Default response</td>
</tr>
<tr>
<td></td>
<td>Valid responses</td>
</tr>
<tr>
<td></td>
<td>For an example of using ADSA to modify a function, see &quot;Step 4: Modify the ADDDEP function&quot; in Section 11, &quot;Modifying the Application Structure Using ADSA&quot;.</td>
</tr>
<tr>
<td>Deleting a function</td>
<td>Display the function definition (if not already displayed) on the Function Definition screen as described above in &quot;Modifying a function&quot; earlier in this table.</td>
</tr>
<tr>
<td></td>
<td>Place a nonblank character next to the Drop prompt</td>
</tr>
<tr>
<td></td>
<td>For an example of using the Function Definition screen, see &quot;Step 6: Further define application functions&quot; in &quot;Defining an Application Structure Using ADSA&quot;.</td>
</tr>
<tr>
<td>Adding a task code</td>
<td>Display the Task Codes screen by selecting 5 from the Main Menu.</td>
</tr>
<tr>
<td></td>
<td>Add the task code: Enter a task code below the Task code heading Enter a function name below the corresponding Function heading.</td>
</tr>
<tr>
<td></td>
<td>For an example of using ADSA to define a task code, see &quot;Step 7: Define a task code&quot; in &quot;Defining an Application Structure Using ADSA&quot;.</td>
</tr>
<tr>
<td>Modifying a task code</td>
<td>Display the Task Codes screen by selecting 5 from the Main Menu.</td>
</tr>
<tr>
<td></td>
<td>Erase the task code from the screen by pressing the ERASE EOF key or typing spaces over the task code.</td>
</tr>
<tr>
<td></td>
<td>For an example of using the Task Codes screen, see &quot;Step 7: Define a task code&quot; in &quot;Defining an Application Structure Using ADSA&quot;.</td>
</tr>
</tbody>
</table>

Notes:
1. To modify an application name:
   a. Add the application definition to the data dictionary, this time using the new name (for instructions, see "Adding an application" earlier in the table) and copy the existing definition using the Copy option under the Add activity on the action bar of the Main Menu.
   b. Delete the old application, as described in "Deleting an application".

2. To add a response process to a dialog, see "Using ADSC" later in this section.

3. Note that ADSA automatically adds a skeleton function with the specified name if the function does not already exist.

4. To modify a response name:
   a. Add the response definition again, this time using the new name (for instructions, see "Adding a response" earlier in the table).
   b. Delete the old response, as described in "Deleting a response" later in the table.

5. To modify a dialog response process, which is a module of process-language code, see "Using ADSC" later in this section.

6. To delete a response process from a dialog, see "Using ADSC" later in this section.

7. For system functions (such as QUIT), you do not need to follow this procedure; system functions are added to the application automatically when you define responses to invoke the system functions.

8. In addition to the listed function types, you also specify the function type for either a user program or menu/dialog function. For more information on these function types, refer to CA ADS Reference Section.

9. Valid also for menu/dialog functions. For more information, refer to CA ADS Reference Section.

10. To modify a function name:
    a. Make a note of specifications for the function on the Function Definition screen.
    b. Delete the function as described in "Deleting a function" earlier in the table.
    c. Add the function using the new name (see "Adding a function" earlier in the table).

Using ADSC

ADSC is the application development tool used to define dialogs. Dialogs are associated with dialog functions in applications. At runtime, a dialog interacts with the user by means of a map (screen). A dialog also performs processing, such as updating the database.

The first screen in an ADSC session is the Main Menu screen. A sample Main Menu screen is shown below:
CA ADS Online Dialog Compiler
CA, Inc.

Type and select. Then Enter or select an action.

Dialog name . . . . . . . . . ______
Dialog version . . . . . . ______
Dictionary name . . . . . ______
Dictionary node . . . . . ______

Screen . . . . . . . . . 1 1. General options
2. Assign maps
3. Assign database
4. Assign records and tables
5. Assign process modules

Enter information on the dialog after prompts in the Specification area.

Select the next ADSC activity or screen from Screen prompt.

To get from one ADSC screen to another, you can either enter the number associated with the screen on the Main Menu (and press [Enter]) or press [PF5] to move through the sequence of screens.

Select an action by tabbing to the action bar or selecting with the command line.

What You Can Do

You can use ADSC to perform the following procedures:

- Adding a dialog
- Modifying a dialog
- Deleting a dialog
- Adding a work record to a dialog
- Modifying a work record in a dialog
- Deleting a work record from a dialog
- Adding a declaration module to a dialog
- Modifying a declaration module specification
- Deleting a declaration module from a dialog
- Adding a premap process to a dialog
- Modifying a premap process specification
Deleting a premap process from a dialog

Adding a response process to a dialog

Modifying a response process specification

Deleting a response process from a dialog

Viewing process module errors

Correcting syntax errors

Correcting discrepancies between dialog components

The following table describes how to use ADSC to perform the procedures listed above. Instructions in the table assume that you have already invoked ADSC, as discussed in Invoking Development Tools (see page 373) earlier in this section.

For more information on these and other ADSC procedures, refer to CA ADS Reference Section.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding a dialog</td>
<td>Enter basic information about the dialog on the dialog Main Menu screen: The dialog name A dictionary name/node (when applicable) <strong>Associate a map with the dialog</strong> on the Map Specifications screen. Associate a subschema or access module with the dialog on the Database Specifications screen.1 If you are creating a skeleton dialog, you can now proceed immediately to Step 5 to compile the dialog. Optionally make additional specifications on appropriate ADSC screens: Specify additional <strong>dialog options</strong> (for example, symbol and diagnostic tables to simplify the application development process) by using the Options and Directives screen. For more information on the Options and Directives screen, see &quot;Step 2: Specify dialog options&quot; in Section12. Associate one or more <strong>work records</strong> or SQL-defined tables with the dialog by using the Records and Tables screen, as described in &quot;Adding a work record to a dialog&quot; later in this table. Associate a <strong>declaration module</strong>, <strong>premap process</strong>, and one or more <strong>response processes</strong> with the dialog by using the Process Modules screen, as described in &quot;Adding processes to a dialog&quot; later in this table. <strong>Create a dialog load module</strong> by selecting the <strong>Compile</strong> activity from the action bar on the Main Menu. For an example of using ADSC to add dialogs, see Section10, &quot;Defining Dialogs Using ADSC&quot;.</td>
</tr>
<tr>
<td>Operation</td>
<td>Procedure</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Modifying a dialog</strong></td>
<td>Display the dialog definition (if not already displayed) by entering the following on the Main Menu screen:</td>
</tr>
<tr>
<td>Procedure to modify a dialog definition. This procedure cannot be used to modify a dialog name2.</td>
<td>The dialog name</td>
</tr>
<tr>
<td></td>
<td>The dictionary name/node (when applicable)</td>
</tr>
<tr>
<td></td>
<td><strong>Modify dialog specifications</strong> on appropriate ADSC screens:</td>
</tr>
<tr>
<td></td>
<td>Optionally add or modify dialog options by using the Options and Directives screen. For more information on the Options and Directives screen, see &quot;Step 2: Specify dialog options&quot; in Section12.</td>
</tr>
<tr>
<td></td>
<td>Optionally add or modify map information on the Map Specifications screen. For more information on the Map Specifications screen, see &quot;Step 3: Name the associated map&quot; in Section9.</td>
</tr>
<tr>
<td></td>
<td>Optionally add or modify subschema or access module information on the Database Specifications screen. For more information on the Database Specifications screen, see &quot;Step 3: Add a subschema&quot; in Section12.</td>
</tr>
<tr>
<td></td>
<td>Optionally add, modify, or delete work record specifications or tables on the Records and Tables screen, as described later in this table.</td>
</tr>
<tr>
<td></td>
<td>Optionally add, modify, or delete the process module specifications on the Process Modules screen, as described later in this table.</td>
</tr>
<tr>
<td></td>
<td>Recompile the dialog load module by selecting the Compile activity on the Main Menu.</td>
</tr>
<tr>
<td></td>
<td>For examples of modifying dialogs, see &quot;Updating modified maps in dialogs using ADSC&quot; in Section11 and &quot;Updating modified process modules in dialogs using ADSC&quot; in Section13.</td>
</tr>
<tr>
<td><strong>Deleting a dialog</strong></td>
<td>Enter the following information on a blank Main Menu screen:</td>
</tr>
<tr>
<td>Procedure to delete a dialog from the data dictionary.</td>
<td>The dialog name</td>
</tr>
<tr>
<td></td>
<td>the dictionary name/node (when applicable)</td>
</tr>
<tr>
<td></td>
<td>Choose the Delete activity from the action bar.</td>
</tr>
<tr>
<td></td>
<td>Confirm or rescind the deletion.</td>
</tr>
<tr>
<td></td>
<td>For an example of using ADSC, see Section10, &quot;Defining Dialogs Using ADSC&quot;.</td>
</tr>
<tr>
<td><strong>Adding a work record to a dialog</strong></td>
<td>Display the Records and Tables screen by selecting4 from the Main Menu screen.</td>
</tr>
<tr>
<td>Procedure to make an existing work record or table available to a dialog and its processes3.</td>
<td>Enter the name of the work record or table to be associated with the dialog under the Record name prompt.</td>
</tr>
<tr>
<td>To add a work record to the data dictionary, see &quot;Using IDD&quot; later in this section.</td>
<td>If this is a work record, enter a nonblank character under the prompt Work.</td>
</tr>
<tr>
<td></td>
<td>For an example of adding a work record to a dialog, see &quot;Step 3: Add a work record&quot; in Section15.</td>
</tr>
<tr>
<td><strong>Deleting a work record from a dialog</strong></td>
<td>Procedure to delete a work record or table from a dialog.</td>
</tr>
<tr>
<td>Procedure does not delete a work record from the data dictionary4.</td>
<td>Display the dialog definition (if not already displayed) by entering the following on the Main Menu screen:</td>
</tr>
<tr>
<td></td>
<td>The dialog name</td>
</tr>
<tr>
<td></td>
<td>The dictionary name/node (when applicable)</td>
</tr>
<tr>
<td></td>
<td><strong>Modify dialog specifications</strong> on appropriate ADSC screens:</td>
</tr>
<tr>
<td></td>
<td>Optionally add or modify dialog options by using the Options and Directives screen. For more information on the Options and Directives screen, see &quot;Step 2: Specify dialog options&quot; in Section12.</td>
</tr>
<tr>
<td></td>
<td>Optionally add or modify map information on the Map Specifications screen. For more information on the Map Specifications screen, see &quot;Step 3: Name the associated map&quot; in Section9.</td>
</tr>
<tr>
<td></td>
<td>Optionally add or modify subschema or access module information on the Database Specifications screen. For more information on the Database Specifications screen, see &quot;Step 3: Add a subschema&quot; in Section12.</td>
</tr>
<tr>
<td></td>
<td>Optionally add, modify, or delete work record specifications or tables on the Records and Tables screen, as described later in this table.</td>
</tr>
<tr>
<td></td>
<td>Optionally add, modify, or delete the process module specifications on the Process Modules screen, as described later in this table.</td>
</tr>
<tr>
<td></td>
<td>Recompile the dialog load module by selecting the Compile activity on the Main Menu.</td>
</tr>
<tr>
<td></td>
<td>For examples of modifying dialogs, see &quot;Updating modified maps in dialogs using ADSC&quot; in Section11 and &quot;Updating modified process modules in dialogs using ADSC&quot; in Section13.</td>
</tr>
<tr>
<td>Operation</td>
<td>Procedure</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Operation</td>
<td>Procedure</td>
</tr>
<tr>
<td>Display the Records and Tables screen by selecting 4 from the Main Menu screen. Enter a nonblank character in the Drop column opposite the work record or table to be deleted. For an example of how to use the Records and Tables screen, see &quot;Step 3: Add a work record&quot; in Section 15.</td>
<td></td>
</tr>
<tr>
<td>Adding process modules to a dialog</td>
<td>Display the Process Modules screen by selecting 5 from the Main Menu. Enter the name of the process module to be associated with the dialog after the Name prompt. Specify the type of process module it is: declaration, premap, response, default response. If this is a response process, enter the following information: A control key (optional) specified after the Key prompt. A response field value (optional) specified after the Value prompt. Note: Each response process must have a key and/or a response field value, or be the default response. Execution status on input errors (optional) enabled by entering a nonblank character before the Execute on edit errors prompt; in this case, the response process is executed at runtime even when input errors are entered. For information on automatic editing, refer to CA IDMS Mapping Facility Section. Press [Enter] to add the process module to the dialog definition. For an example of using ADSC to associate a process module with a dialog, see &quot;Adding process modules to dialogs using ADSC&quot; in Section 12. For more information on the Process Module screen, refer to CA ADS Reference Section.</td>
</tr>
<tr>
<td>Modifying a process module</td>
<td>Display the Process Modules screen. Optionally change the process module name by typing over the displayed name. The new name must identify an existing process module in the data dictionary. Optionally modify specifications: Key Value Execute on errors Type (1=Declaration, 2=Premap, 3=Response, 4=Default response) Press [Enter] to input modified Process Modules screen specifications. For an example of modifying a process module specification, see &quot;Updating modified process modules in dialogs using ADSC&quot; in Section 13.</td>
</tr>
<tr>
<td>Deleting a process module from a dialog</td>
<td>Display the Process Modules screen (if it not already displayed) by selecting 5 from the Main Menu. Enter a nonblank character next to the Drop prompt.</td>
</tr>
<tr>
<td>Operation</td>
<td>Procedure</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>This procedure does not delete the</td>
<td>For an example of using the Process Modules screen, see &quot;Adding process modules to dialogs using ADSC&quot; in Section 12.</td>
</tr>
<tr>
<td>process module from the data dictionary.</td>
<td></td>
</tr>
<tr>
<td><strong>Compiling the dialog</strong></td>
<td><strong>Create a load module for the application</strong> by selecting the <strong>Compile</strong> activity from the action bar on the Main Menu. ADSC will attempt to compile all process modules associated with the dialog and, if successful, create a load module. <strong>Note:</strong> If ADSC finds errors during the compile, the process module is not added to the dialog. ADSC displays an error message. In this case, you must view and correct process module errors and then recompile the process module. For more information, see &quot;Viewing process module errors&quot; later in this table.</td>
</tr>
<tr>
<td>Procedure to compile process code and</td>
<td></td>
</tr>
<tr>
<td>create the dialog load module.</td>
<td></td>
</tr>
<tr>
<td><strong>Viewing process module errors</strong></td>
<td><strong>Display a listing of the module and its errors:</strong> Choose the <strong>View messages</strong> option from the <strong>Compile</strong> activity on the action bar on the Main Menu.</td>
</tr>
<tr>
<td>Procedure to view compile-time errors</td>
<td></td>
</tr>
<tr>
<td>and error messages found by ADSC in a</td>
<td></td>
</tr>
<tr>
<td>process module.</td>
<td></td>
</tr>
<tr>
<td><strong>Correcting syntax errors</strong></td>
<td><strong>Choosing the <strong>View messages</strong> option from the <strong>Compile</strong> activity on the action bar of the Main Menu will bring you to the Compiled Process Modules screen. On the Compiled Process Modules screen, choose <strong>Display</strong> to go to the Dialog Process Source screen. The Dialog Process Source screen will show you the process source and errors encountered by the compiler. Press [PF5] to move to IDD to correct errors. 11 For an example of using IDD and ADSC to correct syntax errors, see &quot;Correct syntax errors&quot; in Section 12.</strong></td>
</tr>
<tr>
<td>Procedure to correct syntax errors (such</td>
<td></td>
</tr>
<tr>
<td>as omitted periods or misspelled words)</td>
<td></td>
</tr>
<tr>
<td>in a process module.</td>
<td></td>
</tr>
<tr>
<td>To view error messages for a process</td>
<td></td>
</tr>
<tr>
<td>module, see &quot;Viewing process module</td>
<td></td>
</tr>
<tr>
<td>errors&quot; earlier in this table.</td>
<td></td>
</tr>
<tr>
<td>**Correcting discrepancies between</td>
<td><strong>Choosing the <strong>View messages</strong> option from the <strong>Compile</strong> activity on the action bar of the Main Menu will bring you to the <strong>Structural Error Display</strong> screen. Correct the errors by using the appropriate development tool:</strong> <strong>For errors made on ADSC screens</strong> (such as an incorrect subschema name), proceed to the appropriate screen and change the specification. <strong>For errors made in a different development tool</strong> (such as a record definition made by using the IDD menu facility): Transfer to the development tool by selecting the <strong>Switch</strong> activity from the action bar on the Main Menu (if you are operating under TCF) and specify the task code for the development tool. For more information on transferring to other development tools, see &quot;Invoking development tools&quot; earlier in this table.</td>
</tr>
<tr>
<td>dialog components**</td>
<td></td>
</tr>
<tr>
<td>Procedure to correct discrepancies</td>
<td></td>
</tr>
<tr>
<td>between a process module and another</td>
<td></td>
</tr>
<tr>
<td>dialog component (for example, a map,</td>
<td></td>
</tr>
<tr>
<td>record, record element, or subschema).</td>
<td></td>
</tr>
<tr>
<td>To view error messages for a process</td>
<td></td>
</tr>
<tr>
<td>module, see &quot;Viewing process module</td>
<td></td>
</tr>
<tr>
<td>errors&quot; earlier in this table.</td>
<td></td>
</tr>
</tbody>
</table>
Operation | Procedure
--- | ---
Correct and recompile the definition:  
To modify maps, see "Using MAPC" later in this section.  
To modify work records, elements, and process modules, see "Using IDD" later in this section.  
**Transfer back to ADSC** (if necessary) by using the **Switch** activity.  
Recompile the dialog.  
For a discussion of correcting discrepancies between dialog components, see "Correct discrepancies" in Section 12.

Notes:

1. You do not have to name the subschema and schema in the following situations:

   If you are creating a skeleton dialog for the basic prototype of an application.
   - If the dialog uses only SQL statements to access the database
   - If the dialog does not access a database

   An access module needs to be named only if the dialog is accessing a database using SQL statements and an existing access module is going to be used.

2. To modify a dialog name, copy the dialog definition to a new dialog name using the **Copy** option of the **Add** activity on the action bar on the Main Menu. Delete the old dialog.

3. Work records associated with the dialog's map are *automatically* available to the dialog and do not need to be added to the dialog separately.

4. To delete a work record from the data dictionary, see "Using IDD" later in this section.

5. To modify process module source statements in the data dictionary, see "Using IDD" later in this section.

6. To delete a process module from the data dictionary, see "Using IDD" later in this section.

7. To define a process module in the data dictionary, see "Using IDD" later in this section.

8. You can associate a response process with several control keys and/or response field values. To do this, add the response process to the dialog several times, each time specifying a different control key and/or response field value for the response process. Only one copy of the compiled response is included in the dialog load module.

9. To modify process module source statements, see "Using IDD" later in this section.

10. Whenever you select the **Compile** activity, ADSC compiles all process modules before creating a load module for the dialog.
When you switch to IDD from ADSC after encountering compile errors, you are in full-screen mode in IDD.

For more information on IDD full-screen mode, refer to *CA IDMS Common Facilities Section*.

**Using MAPC**

MAPC is the application development tool used to define online maps. In CA ADS applications, maps are displayed by dialogs.

The first screen in an MAPC session is the Main Menu screen. A sample Main Menu screen is shown below:

```
Add Modify Compile Delete Display Switch
```

```
CA IDMS/DC Online Map Compiler
CA, Inc.

Map name . . . . . . . . . . _________
Map version . . . . . . . . _____
Dictionary name . . . . _________
Dictionary node . . . . _________
Screen . . . . . . . . . . _
```

1. General options
2. Map-Level help text definition
3. Associated records
4. Layout
5. Field definition

```
Command ==>
Enter F1=Help  F3=Exit  F10=Action
```

You can display an MAPC screen by entering a value opposite the `Screen` prompt and pressing [Enter].

The following table describes how to use MAPC to perform the following procedures:

- Adding a map
- Modifying a map
- Deleting a map
- Adding new fields to a map
- Modifying existing fields on a map
- Deleting fields from a map

For more information on these and other MAPC procedures, refer to *CA IDMS Mapping Facility Section*.

Instructions in the table assume that you have already invoked MAPC, as discussed earlier in this section.
### Operation | Procedure
--- | ---
**Adding a map**
Procedure to add a map definition to the data dictionary
To add an existing map to a dialog, see "Using ADSC" earlier in this section. | **Specific basic information** about the map on the MAPC Main Menu:
The map name
The dictionary name/node (when applicable)
Names the associated database and work records on the Associated Records screen.
Autopaint the map by naming the map fields on the Automatic Screen Painter screen.
Optionally make additional specifications on other MAPC screens.
For more information on available MAPC screens and definition options, refer to *CA IDMS Mapping Facility Section*.
Optionally modify the map layout using the Layout screen.
Create a map load module using the Compile4 activity from the action bar on the Main Menu.
For an example of using MAPC to add a map definition, see Section9, "Defining a Screen Display Using MAPC".

**Modifying a map**
Procedure to modify an existing map.
This procedure cannot be used to modify a map name.
To update a modified map definition in a dialog, recompile the dialog as described in "Using ADSC" earlier in this section. | **Display the map definition** by entering on a Main Menu screen:
The map name
the dictionary name/node (when applicable)
**Optionally add and modify basic information** about the map, including names of associated database and work records.
**Optionally add, modify, and/or delete fields.** You can add, modify, and delete fields at the same time:
To add new fields, see "Adding new fields to a map" later in this table.
To modify fields, see "Modifying existing fields on a map" later in this table.
To delete fields, see "Deleting fields from a map" later in this table.
Optionally add, modify, and/or delete specifications on other MAPC screens.
For more information on available MAPC specifications, see *CA IDMS Mapping Facility Section*.
**Recompile the map load module** by specifying the Compile4 activity from the action bar on the Main Menu.
**Note:** If new copies of maps are not automatically loaded in the program pool at your site, return to CA IDMS/DC or DC/UCF and issue the following command before executing the modified map:
DCMT VARY PROGRAM modified-map-name NEW COPY.
This command causes a new copy of the map to be loaded in the program pool.
For an example of using MAPC to modify a map, see Section12, "Modifying a Map Using MAPC".

**Deleting a map**
Procedure to delete a map from the data dictionary.
To delete a map from a dialog, modify the dialog as described in "Using ADSC" earlier in this section. | **Identify the map** to be deleted on the Main Menu screen by entering:
The map name
the dictionary name/node (when applicable)
Choose the Delete action from the action bar
Confirm or rescind the action
The next time you modify a dialog associated with a deleted map, ADSC displays a message warning you that the map has been
### Operation

<table>
<thead>
<tr>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleted. Users can continue to execute a deleted map until a new copy of the map is loaded in the program pool, provided that automatic load is not specified in the system generation OLM statement. For an example of using MAPC, see Section 9, &quot;Defining a Screen Display Using MAPC&quot;.</td>
</tr>
</tbody>
</table>

### Adding new fields to a map

Procedure to add new fields to a map.

To move or modify fields, see "Modifying existing fields on a map" later in this table. To delete fields, see "Deleting fields from a map" later in this table.

**Display the Layout screen** from the Main Menu by entering 2 opposite the Screen prompt and pressing [Enter].

**Define each new field** on the Layout screen as described below:

- **Move the cursor** to the screen position that immediately precedes the starting position of the field.
- **Type a start-field character** to signal the start of the field. The field starts in the column immediately after the start-field character.
- **For a literal field only**, specify the value to be displayed at runtime by typing a literal string after the start-field character, as shown below for a field that displays the value DEPARTMENT INFORMATION:
  
  ```
  ;DEPARTMENT INFORMATION
  ```

**Note:** For a variable field, you will specify the value to be displayed at runtime in Step 3 later in this procedure.

- Optionally add additional new fields as described in Steps 1, 2, and 3 above.
- Press [Enter] to save the current map layout for further definition.

**Select fields for further editing** by positioning the cursor on the field and pressing [PF2] or by overtyping the start-field character with the field-select character (the default is a percent sign).

**Optionally override default specifications** for the variable field:
For an example of adding fields to a map, see Section 9, "Defining a Screen Display Using MAPC".

### Editing fields

Procedure to edit variable fields on a map.

Press [PF5] from the Layout screen to access the Literal or Field Definition screen.

**Edit variable fields** on the Field Definition screen as described below:

- **Specify the value to be displayed by the field** by naming a record element or system-supplied field ($RESPONSE, $MESSAGE, or $PAGE) after the Element prompt.
- **Optionally override default specifications** for the variable field:
  
  For an example of adding fields to a map, see Section 9, "Defining a Screen Display Using MAPC".

### Modifying existing fields on a map

Procedure to modify or move existing fields on a map.

Adding new fields to a map is presented in "Adding new fields to a map" earlier in this table.

To associate the modified map with any dialogs that use the map, you recompile the dialog load module. See "Using ADSC" earlier in this section.

**Display the Layout screen** from the Main Menu by entering 4 at the Screen prompt.

**Use the alternate set of function keys to mark and move the fields**

- **Select one or more fields to be modified** by positioning the cursor on the field and pressing [PF2] or by overtyping the start-field character with a percent sign.
- **Modify each field definition selected** as described below:
  
  Display the Field Definition or Literal Definition screen.
  Modify any specifications for the field.

- **Recompile the map load module** if there are no more updates to be made to the map by selecting the Compile 4 activity from the action bar on the Main Menu.

For an example of modifying fields on a map, see Section 12, "Modifying a Map Using MAPC".
**Deleting fields from a map**

Procedure to delete existing fields from a map

To associate the modified map with any dialogs that use the map, recompile the dialog load module. See "Using ADSC" earlier in this section.

**Procedure**

1. Display the Layout screen by entering 4 at the Screen prompt on the Main Menu.
2. Make sure that insert-character mode is disabled by pressing the RESET key.
3. Select the fields to be deleted by positioning the cursor on the field or fields and pressing [PF2] or by overtyping the start-field character with the field-select character.
4. Press [PF5] to move to the Literal Definition or Field Definition screen.
5. Enter a nonblank character next to the Drop field prompt to delete the field.
6. Recompile the map load module (if there are no more updates to be made to the map) by selecting the Compile activity from the action bar on the Main Menu.

**Notes:**

1. To modify a map name:
   - Make a copy of the original map, using the new name for the copy. For more information on copying a map, refer to CA IDMS Mapping Facility Section.
   - Create a load module for the new copy.

2. The start-field character is defined at system generation time; for example, default start-field characters are the field mark (;) or the left brace ({}).

3. The select-field and start-field characters are defined at system-generation time and can vary from site to site; the default select-field character is the percent sign (%).

4. The dictionary is updated only on a Compile or a Delete action. A record becomes map-owned only after the map is compiled.

**Using the IDD Menu Facility**

As an application developer, you can use the IDD menu facility to define data (records and elements) and process modules in the data dictionary.

Alternatively, you can use online IDD to define data and process modules.

For information on how to use online IDD, see the CA IDMS IDD Quick Reference Section.

The first screen in an IDD menu facility session is the Master Selection screen. A sample Master Selection screen is shown below:
DICTIONARY NAME........: DEMO  NODE NAME....:
USER NAME............:  PASSWORD............:
USAGE MODE..........: X UPDATE _ RETRIEVAL
PFKEY SIMULATION...: X OFF _ ON

- ATTR = ATTRIBUTE <PF2> - PROC = PROCESS <PF3>
- CLAS = CLASS <PF4> - PROG = PROGRAM <PF5>
- ELEM = ELEMENT <PF6> - RECD = RECORD <PF7>
- FILE = FILE <PF8> - TABL = TABLE <PF9>
- MODU = MODULE <PF10> - USER = USER <PF11>
- ENTL = USER DEFINED ENTITY LIST _ SYST = SYSTEM
- MSGS = MESSAGE _ OPTI = OPTIONS
- QFIL = QFILE _ DISP = DISPLAY ALL
- RECD = RECORD _ HELP = HELP <PF1>

To get from one IDD screen to another, you can enter the identifier for the next screen in the current screen's command area, select the screen identifier from the Activity Selection area, or press the control key associated with the screen. Identifiers and control keys for available screens are listed in the Activity Selection area of IDD menu facility screens.

The following table describes how to use the IDD menu facility to perform the following procedures:

- Adding an element
- Modifying an element
- Deleting an element
- Adding a work record
- Modifying a work record
- Deleting a work record
- Adding a process module
- Modifying a process module
- Copying a process module
- Deleting a process module

For more information on using the IDD menu facility, refer to CA IDMS Common Facilities Section.

For information on other operations you can perform by using IDD, refer to CA IDMS IDD Quick Reference Section.

Instructions in table assume that you have already invoked IDD, as discussed earlier in this section.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
</table>
| **Adding an element** | Procedure to add a new element to the data dictionary  
To add an existing element to a work record, see "Adding a work record" later in this table.  
**Display the Element Entity screen** by entering the identifier (elem) for the screen in the command area, as shown below:

```
- elem
```

**Define the element** on the Element Entity screen:  
Type an element name.  
Select the ADD action and deselect (space over) DISPLAY.  
Optionally type a description of the element.  
Specify a picture; for example, PIC X(20).  
Select a usage mode; for example, COMP-3 (default is DISPLAY).  
For an example of using the IDD menu facility to define an element, see "Step 1: Define an element" in Section 14. |
| **Modifying an element** | Procedure to modify an element definition in the data dictionary.  
To update the modified element in any records that contain the element, see "Modifying a work record" later in this table.  
This procedure cannot be used to modify an element name.  
**Display the element to be modified** on the Element Entity screen by typing the identifier (ELEM) for the screen in the command area, followed by the name of the element, as shown below for an element named ELEMENT1:

```
- elem element1
```

**Modify the element:**  
Select the MODIFY action and deselect DISPLAY.  
Modify any of the following specifications:  
Description  
Picture  
Usage mode  
For an example of using the Element Entity screen, see "Step 1: Define an element" in Section 14. |
| **Deleting an element** | Procedure to delete an element definition from the data dictionary.  
This procedure cannot be used to delete an element that already belongs to a work record.  
**Display the element to be deleted** on the Element Entity screen by typing the identifier (ELEM) for the screen in the command area, followed by the name of the element, as shown below for an element named ELEMENT1:

```
- elem element1
```

Delete the element:  
Select the DELETE action.  
Deselect the DISPLAY action.  
For an example of using the Element Entity screen, see "Step 1: Define an element" in Section 14. |
| **Adding a work record** | Procedure to add a work record to the data dictionary.  
To add elements to the data dictionary, see "Adding an element" earlier in this table.  
**Display the Record Entity screen** by entering the identifier (recd) for the screen in the command area:

```
- recd
```

**Define the work record** on the Record Entity screen:  
Type a record name.  
Select the ADD action and deselect DISPLAY.  
Optionally make additional specifications for the work record.  
For more information on available record options and screens, refer to **CA IDMS IDD Quick Reference Section**.  
Associate existing elements with the work record by using the Record Element screen: |
<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display the Record Element screen from the Record Entity screen by entering the identifier (relm) for the screen in the command area:</td>
<td></td>
</tr>
<tr>
<td>relm</td>
<td></td>
</tr>
<tr>
<td>Associate an element with the work record:</td>
<td>Type the name of an existing element.</td>
</tr>
<tr>
<td>Optionally override any existing element specifications for:</td>
<td>Picture</td>
</tr>
<tr>
<td>Usage mode</td>
<td></td>
</tr>
<tr>
<td>Optionally make other specifications for the element.</td>
<td></td>
</tr>
<tr>
<td>For more information on available record options and screens, refer to CA IDMS IDD Quick Reference Section.</td>
<td></td>
</tr>
<tr>
<td>Optionally associate another element with the work record as described in Step 3.2 above, after first pressing the page-forward key (default is [PF8]) to display a blank Record Element screen.</td>
<td></td>
</tr>
<tr>
<td>For an example of using the Element Entity screen, see &quot;Step 2: Define a work record&quot; in Section 14.</td>
<td></td>
</tr>
</tbody>
</table>

**Modifying a work record -- continued on next two pages**

Procedure to modify a work record in order to replace elements in the record or to modify specifications for record elements. This procedure cannot be used to modify a work record name.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display the work record to be modified by typing the identifier (RECD) for the Record Entity screen in the command area, followed by the name of the record, as shown below for a record named WK-RECORD1:</td>
<td></td>
</tr>
<tr>
<td>recd wk-record1</td>
<td></td>
</tr>
<tr>
<td>Optionally add, modify, or delete work record specifications.</td>
<td>For more information on record specifications and screens, refer to CA IDMS IDD Quick Reference Section.</td>
</tr>
<tr>
<td>Optionally modify element specifications for the record:</td>
<td></td>
</tr>
<tr>
<td>To associate an element with the work record:</td>
<td>Display the Record Element screen by entering the identifier (relm) for the screen in the command area:</td>
</tr>
<tr>
<td>relm</td>
<td></td>
</tr>
<tr>
<td>Associate an element with the record:</td>
<td>Type the name of an existing element.</td>
</tr>
<tr>
<td>Optionally override any existing element specifications for:</td>
<td>Picture</td>
</tr>
<tr>
<td>Usage mode</td>
<td></td>
</tr>
<tr>
<td>Optionally make other specifications for the element.</td>
<td>For more information on record element specifications, refer to CA IDMS IDD Quick Reference Section.</td>
</tr>
<tr>
<td>Optionally associate another element with the record as described in Step b above, after first pressing the page-forward key (default is [PF8]) to display a blank Record Element screen.</td>
<td></td>
</tr>
<tr>
<td>To replace an element in the record:</td>
<td>Display the record element specification in either of the following ways:</td>
</tr>
<tr>
<td>When the record contains a few record elements:</td>
<td>Display the Record Element screen by entering the identifier (relm) for the screen in the command area:</td>
</tr>
</tbody>
</table>
Modifying a work record -- continued

When the record contains several record elements:

Display the Record Element List screen by entering the identifier (rell) for the screen in the command area:

rell

From the displayed list of elements, select the record element to be replaced by entering a nonblank character in the SELECT column for the element.

Modify the record element specification:

Select the REPLACE action.

Note: The REPLACE action:
- Deletes the element from the work record
- Adds a new copy of the element to the record.
- REPLACE does not modify the original element in the data dictionary.

Optionally type a new element name to replace the existing element with another.

Optionally override any existing element specifications for:
- Picture
- Usage mode
- Optionally modify other specifications

For more information on available record element specifications, refer to CA IDMS IDD Quick Reference Section.

Optionally select and modify another record element specification:

Return to the Record Element List screen by pressing CLEAR (after pressing [Enter] to make sure that all record element specifications are entered).

Select the record element to be modified by entering a nonblank character in the SELECT column for the element.

Modify the record element as described in Step b above.

To remove an element from the record:

Display the record element specification:

Display the Record Element List screen by entering the identifier (rell) for the screen in the command area:

rell

From the displayed list of elements, select the record element specification to be removed by entering a nonblank character in the SELECT column for the element.

Remove the element by selecting the REMOVE action.

Note: The REMOVE action deletes the record element specification from the record but not from the data dictionary.

For an example of using the IDD menu facility for records, see Section15, "Defining Work Records Using IDD".

Page through record elements in the record (if necessary) to display the required record element. To do this, press the page-forward key (default is [PF8]).
### Operation | Procedure
---|---
**Deleting a work record**
Procedure to delete a record from the data dictionary. This procedure cannot be used to delete a work record used by a map or a dialog. This procedure does not delete record elements from the data dictionary.
- **Display the work record to be deleted** on the Record Entity screen by typing the identifier (RECD) for the screen in the command area, followed by the name of the record to be deleted, as shown below for a record named WK-RECORD1:
  
  ```
  recd wk-record1
  ```

- **Delete the work record**: Select the DELETE action. Deselect the DISPLAY action.
  
  For an example of using the IDD menu facility for work records, see [Defining Work Records Using IDD](#).

**Adding a process module**
Procedure to add a process module definition and source statements to the data dictionary.
To make an existing process module the premap or response process for a dialog, see "Using ADSG" earlier in this section.
- **Display the Process Entity screen** by entering the identifier (PROC) for the screen in the command area:
  
  ```
  proc
  ```

- **Specify basic information on the process module on the Process Entity screen**: Type the process module name.
  Select the ADD action and deselect DISPLAY.

  Optionally type a description for the process module.

  **Display the Process Source screen** from the Process Entity screen by entering `srce` in the command area:
  
  ```
  srce
  ```

- **Enter process source statements for the module**: Type one or more lines of process statements on a page of the Process Source screen.

  **Note**: Do not extend process source statements beyond column 72.

  Optionally enter additional pages of process source statements. For each new page:
  
  - Place the cursor on the line after which new source statements are to be inserted.
  - Press [PF4] (default) to open new lines after the cursor.
  - Enter lines of process statements on the new page.
  - Press [PF5] (default) to apply your changes to the work file maintained by the IDD menu facility.

  **Press [Enter]** to add the process source statements to the process module in the data dictionary.

  For an example of using the IDD menu facility to define a process module, see "Defining process modules using IDD" in Section 12. A process module’s source statements are compiled when the process module is added to a dialog. For information on adding process modules (as premap and response processes) to a dialog, see "Using ADSC" earlier in this section.

**Modifying a process module**
Procedure to modify a process module in the data dictionary. This procedure does not update dialogs that use the process.
- **Display the process module to be modified** on the Process Entity screen by typing the identifier (PROC) for the screen in the command area, followed by the name of the process module, as shown below for a process module named PROCESS1:
### Operation

**Copying a process module**

Procedure to copy source statements from one process module to a new process module.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copying a process module</td>
<td>Display the Process Entity screen by entering the identifier (proc) for the Process Entity screen in the command area:</td>
</tr>
<tr>
<td></td>
<td>proc</td>
</tr>
<tr>
<td></td>
<td>Specify basic information on the new process module on the Process Entity screen:</td>
</tr>
<tr>
<td></td>
<td>Type the name of the new process module.</td>
</tr>
<tr>
<td></td>
<td>Select the ADD action and deselect DISPLAY.</td>
</tr>
<tr>
<td></td>
<td>Optionally type a description for the new process.</td>
</tr>
<tr>
<td></td>
<td>Display the Copy screen by entering the identifier (copy) for the screen in the command area:</td>
</tr>
<tr>
<td></td>
<td>copy</td>
</tr>
<tr>
<td></td>
<td>Enter process module specifications:</td>
</tr>
<tr>
<td></td>
<td>Type the name of the original process module after the COPY FROM PROCESS NAME prompt.</td>
</tr>
<tr>
<td></td>
<td>Select the PROCESS TEXT action to copy the process commands in the original process module to the current process module.</td>
</tr>
<tr>
<td></td>
<td>Modify process statements if necessary, by using the Process Source screen as described in &quot;Modifying a process module&quot; earlier in this table.</td>
</tr>
<tr>
<td></td>
<td>For an example of using IDD for process modules, see Section13 &quot;Adding Process Logic to a Dialog&quot;.</td>
</tr>
</tbody>
</table>

### Procedure

**module9.**

This procedure cannot be used to modify a process module name10.

**proc process1**

Modify process statements, as necessary:
Display process statements for the module on the Process Source screen by entering the identifier (srce) for the screen in the command area:

```
srce
```

Add, modify, and delete process statements. To insert one or more lines of statements:
Place the cursor on the line after which new source statements are to be inserted.
Press [PF4] (default) to open new lines after the cursor.
Enter lines of process statements on the new page.
Press [PF5] (default) to apply the new statements to the work file maintained by the IDD menu facility.

**Note:** Do not extend statements beyond column 72.
Optionally page through the Process Source screen:
Press [PF8] (default) to page forward.
Press [PF7] (default) to page backward.
Press [Enter] to modify the process module stored in the data dictionary.
For an example of modifying a process module, see Section14, "Modifying Process Logic in a Dialog".

**Deleting a process module**

Procedure to delete a process module from the data dictionary.
To delete a process module (premap or response process) from a dialog, see "Using ADSC" earlier in this section.

**Procedure**

Display the process module to be deleted by typing the identifier (PROC) for the Process Entity screen in the command area, followed by the name of the process module, as shown below for a process module named PROCESS1:

```
proc process1
```

Delete the process module:

Select the DELETE action.
Deselect the DISPLAY action.

For an example of using IDD for process modules, see Section 13 "Adding Process Logic to a Dialog".

**Notes:**

1. You can also display any IDD menu facility screen by using either of the following methods:
   - Select the activity that identifies the screen (in this case, ELEM) from the Activity Selection area of the current screen and press [Enter].
   - Press the control key listed for the screen (in this case, [PF6]).

2. To modify an element name:
   - Add the element definition to the data dictionary again, this time using the new name. For instruction, see "Adding an element", earlier in the table.
   - Add the new element to and remove the old element from any work records that contain the element, as described in "Modifying a work record" later in the table.
   - Then delete the old element from the data dictionary, as described below in "Deleting an element".

3. To delete an element that belongs to a work record:
   - Remove the element from the record, as described in "Modifying a work record" later in the table.
   - Delete the element from the data dictionary.

4. To modify a work record name:
   - Add the work record definition to the data dictionary again, this time using the new name. For instruction, see "Adding an element", earlier in the table.
   - Delete the old record from the data dictionary as described in "Deleting a work record" later in the table.

5. If the element already belongs to the record, specifications on the Record Element screen modify previous specifications for the record element.
If the work record participates in maps and/or dialogs:

- Delete the record from each map (by using the MAPC Associated Records screen) and/or from each dialog (by using the ADSC Records and Tables screen).
- Delete the record from the data dictionary.

To delete elements from the data dictionary, see "Deleting an element" earlier in the table.

For information on specific process statements, refer to CA ADS Reference Section.

To update dialogs that use the module, see "Modifying a premap process specification" or "Modifying a response process specification" as appropriate, in "Using ADSC" earlier in this section. At this point, the process module's source statements are compiled.

To modify a process module name:

- Add a new process module, using the new name, on the Process Entity screen.
- Copy the source statements from the original process module by using the Copy screen, as described in "Copying a process module" later in the table.
- If appropriate, delete the original process module from the data dictionary, as described in "Deleting a process module" later in the table.

You can also use one process module's source statements in another process module by using the INCLUDE command. The INCLUDE command names the process module whose source statements are included in the current process module at compile time.

Layout Of the DEPARTMENT Record

This section presents the layout and configuration of the sample DEPARTMENT record. DEPARTMENT is defined in the non-SQL defined demonstration database that can be installed with the system.

Definition

As defined at installation time:

- **The DEPARTMENT record owns the EMPLOYEE record.**
  In order to delete a department that owns employees, you must either disconnect or delete the department’s employees. To delete a department along with its employees, you use a process command like:
  ```
  ERASE DEPARTMENT ALL MEMBERS.
  ```

- **The DEPARTMENT record is defined in the ORG-DEMO-REGION.**

Sample DEPARTMENT Record Layout
What is below shows the layout of the DEPARTMENT record as defined at installation time.

⚠️ **Note:** The demonstration database and the DEPARTMENT record may be defined differently at your site.

**Record:** DEPARTMENT   **Version:** 100

**Location mode:** CALC  
**CALC field...:** DEPT-ID-0410  
**Duplicates records are not allowed**

**Element:** DEPT-ID-0410  
**Picture:** 9(4)  
**Usage...:** Display

**Element:** DEPT-NAME-0410  
**Picture:** X(45)  
**Usage...:** Display

**Element:** DEPT-HEAD-ID-0410  
**Picture:** 9(4)  
**Usage...:** Display

---

**Using ADS Trace**

The online trace of a dialog, or of specific processes within a dialog, provides a convenient, efficient method of searching for errors. The flexible organization of CA ADS Trace allows comprehensive tracing or selective tracing of:

- Control commands
- Specified element values
- Database verbs
- DEFINE SUBROUTINE statements
- A specified segment of a process.

CA ADS Trace adds the necessary statements to the dialog code automatically. After the traced dialog is executed, a replay can be viewed at a terminal, printed, or moved to a special queue.

During execution of an CA ADS dialog, CA ADS Trace produces an online trace that can be replayed as often as necessary. By using this trace utility, programmers and application developers can pinpoint the causes of dialog errors right at their terminals. For more information, see the following topics:

- Online Trace (see page 400)
- CA ADS Trace Access (see page 403)
- Trace Generation Overview (see page 418)
- ADS Trace Replay (see page 433)
Online Trace

Building the Trace

Before a specific dialog is executed, you can select trace options from a special screen. Using those options, CA ADS Trace automatically enters the trace code in the dialog.

Replaying the Trace

After the dialog is recompiled and executed with the selected trace options, a trace replay can be viewed, printed, or saved in a queue. If the dialog execution terminates abnormally, the replay shows the sequence of execution right up to the termination; CA ADS Trace preserves the sequence in the replay, without any rollback.

CA ADS Trace traces not only the specified dialog, but also all subroutines through which dialog control passes. While the dialog is executing, the Trace option captures and highlights information on DML command processing and the DML error status. It can also trace the execution-time contents of any exhibited dialog work record, database field, or map variable.

For more information, see the following topics:
- Environment (see page 400)
- Flexibility (see page 400)
- Typical Session (see page 402)

Environment

CA ADS Trace operates under CA IDMS and CA ADS. It provides full-screen display on all IBM 3270-type terminals.

Flexibility

Contents
- Easy Selection of Options (see page 401)
- Trace Options (see page 401)
- Replay Options (see page 401)
- View Replay Options (see page 402)

CA ADS Trace is a flexible tool. It allows tracing of a particular dialog without affecting the execution of that dialog at another terminal. You can easily make a variety of choices.
Initially, you select trace options for trace generation. After the traced dialog is executed, you select replay options. Then, you can use several commands and PF keys to select options for viewing the replay.

**Easy Selection of Options**

You make selections from menus and formatted selection screens. PF keys facilitate moving to other functional screens, or paging up and down through a series of screens.

**Trace Options**

When specifying trace options, you can:

- Direct CA ADS Trace to enter the trace code automatically, by entering a single letter next to the process name on the Build Trace Code screen.
- Exclude control commands from the trace.
- Trace only DEFINE SUBROUTINE commands. (After reviewing the trace replay that shows the DEFINE SUBROUTINE commands, you can decide whether or not to trace the whole dialog or part of the dialog, and which options to use.)
- Direct CA ADS Trace to display the execution-time contents of any exhibited dialog work records, database fields, or map variables.
- Direct the Trace option to display user-supplied literal statements so that specific events in the trace replay will be highlighted.
- Direct the Trace option to trace a segment of the dialog (internal trace), marked by TRACE ON and TRACE OFF statements.
- Select individual process modules for tracing (and exclude other process modules). This feature is especially useful in a complex transaction with many transfers of control.
- Trace only database verbs. After each database verb, the trace replay shows the error status of the verb.
- Limit the number of lines to be stored in the scratch area.
- Specify a Generate Wait Interval to control the number of times the interrupt routine is called during module trace generation.
- Review the status of existing traced processes and dialogs.
- Remove trace statements from a process or dialog.

**Replay Options**

Once the dialog is traced, you can:

- Display the trace replay on the screen.
Print the trace replay.

Move the trace replay into a queue for later observation. On the Move Replay screen, you can enter a 20 character description indicating the purpose of the trace, or other information, to help identify which trace it is.

Delete the trace replay.

Recall a replay from the queue, for viewing or printing.

View Replay Options

The system displays the trace replay on the screen in pages that each contain 15 lines of trace data. You can:

- Scroll up and down among the pages.
- Search for a particular string of characters.
- Skip up or down any specified number of lines.
- Switch directly to the Move, Print, or Delete screen.

Typical Session

In a typical session, a programmer might be trying to debug a dialog that terminates abnormally when executed. The sequence of events shown in Figure 1.1 illustrates a session where the programmer uses CA ADS Trace to help find the errors in a dialog that terminated abnormally.

The programmer wants to see the values of designated elements exhibited when the trace is replayed, and also wants special literal statements displayed in the replay for quick reference. The programmer can add exhibit and literal statements to the code before accessing CA ADS Trace by using CA IDMS DME.

When CA ADS Trace is accessed in the typical session described here, the first choice is Build Trace Code. On the Build Trace Code screen, the programmer names the dialog and selects the trace options.

When the options have been entered, the system begins automatically placing trace code in the dialog. A confirmation message in the message area signals that the trace code has been built. After the trace code is in the dialog, the programmer adds a special work record to the dialog, and then recompiles and executes it.

When the execution has terminated (in this example, abnormally), a replay of the trace is ready and is listed on the Select Replay screen of CA ADS Trace. The programmer scrolls through the replay to find the cause of termination.

When a TRACE OFF option is entered on the Build Trace Code screen, CA ADS Trace automatically removes the trace statements. To correct errors in the dialog, the programmer uses CA IDMS DME.
The programmer must then remove the special work record from the dialog. When this step is complete, the corrected dialog is recompiled and ready for normal execution. The following figure shows a typical CA ADS Trace session.

Typical Session

CA ADS Trace Access
To access the CA ADS Trace option, enter task code ADST at the System Prompt on the system's main screen.

- Entry Screen (see page 404)
- Common Fields on Functional Screens (see page 404)
- Main Menu (see page 405)

**Entry Screen**

Upon system access, CA ADS Trace displays the CA ADS Trace Entry screen. To proceed to the Main Menu, press the ENTER key.

```
CCCCCCCCC
CCCCCCCCC
CCC
CCC  AAAA
CCC  AAAAA
CCC  AAAA
CCC  AAA  AAA
CCAAACCCCC
AAACCCCCCCC
AAA  AAA
AAA  AAA
AAA  AAA
PRESS ENTER TO RECEIVE MAIN MENU
```

**Common Fields on Functional Screens**

Several fields are common to all functional screens in CA ADS Trace. Here are descriptions of the fields designated by numbers in the following screen.

**Screen title**

**Current time** -- In the format hh:mm:ss, where the time is on a 24-hour clock.

**Current date** -- In the format mm/dd/yy.

**Message**

**area** -- CA ADS Trace messages appear in the third line of the screen. Complete explanations of all messages are in Section6, Messages.
Main Menu

The Main Menu fields are described as follows:

**OPTION** -- The field in which to enter the number or letter of a menu selection.

**DICTIONARY** -- The name of the dictionary to which you are signed on. The first time the Main Menu is displayed, this field is blank. You can enter a dictionary name or use the default, which is the Primary dictionary. If you return to the Main Menu during a session, you can access another dictionary by typing over the dictionary name in this field.

**NODE** -- The DDS node in which the dictionary resides. If you enter the name of a secondary dictionary, be sure that the node and dictionary correspond.

**Menu Selections** -- Numbers and descriptions of the available options. In the OPTION field, you can enter any of the numbers 1 to 4, corresponding to the listed selections. To leave CA ADS Trace from the Main Menu, enter an X in the OPTION field or press CLEAR.

```
OPTION ===> DICTIONARY ===> ACCT
  1 - BUILD TRACE CODE  NODE ===> NODE
  2 - SELECT REPLAY
  3 - CREATE REPLAY FROM QUEUE
  4 - REVIEW TRACE STATUS
  X - EXIT ADS/TRACE
```

For more information, see the following topics:
- Conceptual View of CA ADS Trace (see page 405)
- Program Function Keys (see page 407)
- Trace Generation (see page 409)
- Trace Replay (see page 412)

Conceptual View of CA ADS Trace

This section presents an overview of the CA ADS Trace organization and the PF keys. It then describes a typical session, including compiling the traced dialog, executing the dialog, replaying the trace, and removing the trace code.

- Building a Trace (see page 406)
- Recompiling and Executing a Dialog (see page 407)
- Replaying the Trace (see page 407)
- Removing the Trace (see page 407)

CA ADS Trace performs two important functions—building the trace statements into the source code of the processes in a dialog, and replaying the trace after the dialog is executed. The following diagram outlines these functions.
Building a Trace

You can start automatic entry of trace statements into the source code by selecting an option from those listed on the Build Trace Code screen. (Before the trace code is built, you can add LITERAL or EXHIBIT statements to the source code. You can also add internal trace statements if you want to trace only a segment of a process.)
The Review Trace Status screen presents a record of which processes have traces turned on or off and what types of traces were applied to each process. From the Review Trace Status screen, you can position the cursor to select a process to go back to the Build Trace Code screen.

Recompiling and Executing a Dialog

After the trace statements are in place in the process source code, leave CA ADS Trace and enter the CA ADS Dialog Compiler (ADSC) to add a work record, recompile the dialog, and execute the dialog. When the dialog has executed successfully or terminated abnormally, a replay of the trace (up to the point of termination) is available back in CA ADS Trace.

Replaying the Trace

The Select Replay option from the Main Menu allows you to select viewing the replay at the terminal, moving it to a special queue, printing it, or deleting it.

Once a replay has been moved to the queue, the Create Replay from Queue screen allows a replay to be brought back to the active replay area. The Create Replay from Queue screen displays a list of the replays currently in the queue. You can select one of the replays to move to the Select Replay option screen, or delete one or more replays.

Removing the Trace

After reviewing the trace replay, return to the Build Trace Code screen and select TRACE OFF options, which direct CA ADS Trace to remove the trace statements from the source code. Finally, return to ADSC to remove the work record and recompile the dialog.

Program Function Keys

Program Function (PF) keys provide easy movement to another level or to scroll within a series of screens with more than one screen of data. PF3 through PF8 are not listed on the screens. The functions are described in following table and diagram.

<table>
<thead>
<tr>
<th>Key</th>
<th>Meaning Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEA</td>
<td>Exit to CA-IDMS.</td>
</tr>
<tr>
<td>EXIT</td>
<td></td>
</tr>
<tr>
<td>PF1</td>
<td>Review or Status</td>
</tr>
<tr>
<td>PF2</td>
<td>Build or Code</td>
</tr>
<tr>
<td>PF13</td>
<td></td>
</tr>
<tr>
<td>PF14</td>
<td></td>
</tr>
</tbody>
</table>

END
<table>
<thead>
<tr>
<th>Key</th>
<th>Meaning Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF3</td>
<td>Exit to the next higher level.</td>
</tr>
<tr>
<td>or PF15</td>
<td>Select Replay from Delete Replay, Print Replay, Move Replay, or View Replay.</td>
</tr>
<tr>
<td></td>
<td>To Create Replay from Confirm Delete.</td>
</tr>
<tr>
<td>PF4</td>
<td>RETURN</td>
</tr>
<tr>
<td>or PF16</td>
<td>Return to the CA ADS Trace Main Menu.</td>
</tr>
<tr>
<td>PF5</td>
<td>TOP</td>
</tr>
<tr>
<td>or PF17</td>
<td>In a series of screens, go to the first page of the current display.</td>
</tr>
<tr>
<td>PF6</td>
<td>BOTTOF</td>
</tr>
<tr>
<td>or PF18</td>
<td>In a series of screens, go to the last page of the current display.</td>
</tr>
<tr>
<td>PF7</td>
<td>UP</td>
</tr>
<tr>
<td>or PF19</td>
<td>Move the &quot;window&quot; one page toward the first page of the current display. CA ADS</td>
</tr>
<tr>
<td></td>
<td>Trace replay screens indicate the current page and the number of pages in the</td>
</tr>
<tr>
<td></td>
<td>series. Also used with SKIP and SEARCH commands.</td>
</tr>
<tr>
<td>PF8</td>
<td>DOWN</td>
</tr>
<tr>
<td>or PF20</td>
<td>Move the &quot;window&quot; one page toward the last page of current display. CA ADS Trace</td>
</tr>
<tr>
<td></td>
<td>replay screens indicate the current page and the number of pages in the series.</td>
</tr>
<tr>
<td></td>
<td>Also used with SKIP and SEARCH commands.</td>
</tr>
<tr>
<td>PF9</td>
<td>Select Replay</td>
</tr>
<tr>
<td>or PF21</td>
<td>Jump to Select Replay from Create Replay.</td>
</tr>
<tr>
<td>PF10</td>
<td>Move Replay</td>
</tr>
<tr>
<td>or PF22</td>
<td>Jump to Move Replay from View Replay or from Print Replay.</td>
</tr>
<tr>
<td>PF11</td>
<td>Print Replay</td>
</tr>
<tr>
<td>or PF23</td>
<td>Jump to Print Replay from View Replay.</td>
</tr>
<tr>
<td>PF12</td>
<td>Delete Replay</td>
</tr>
<tr>
<td>or PF24</td>
<td>Jump to Delete Replay from View Replay.</td>
</tr>
</tbody>
</table>
Trace Generation

Contents

- Adding EXHIBIT and LITERAL Statements (see page 410)
- Building Trace Code (see page 410)
This section describes a typical sequence of events for trace generation. The events of the sequence are explained in more detail on the following pages. This is only one possible sequence. For descriptions of all possibilities, see Trace Generation (see page 409).

Adding EXHIBIT and LITERAL Statements

Before entering CA ADS Trace, add LITERAL and EXHIBIT statements at appropriate places in the source code of a specific process. The IDD Online Compiler can be used for this task (see the CA IDMS IDD DDDL Reference Section) Also, if you have installed the product, CA IDMS DME is an efficient tool for entering the statements.

Building Trace Code

Next, from the CA ADS Trace Main Menu, the user selects option 1, Build Trace Code. For this sample sequence, the dialog selected for tracing is in the primary dictionary, which is the default dictionary.

On the Build Trace Code screen, the user enters the dialog name. CA ADS Trace then redisplays the screen, showing the first ten processes of the specified dialog. The dialog version number defaults to 0001 because no version number was entered.

In the Trace On column, the user enters trace options for various processes. CA ADS Trace automatically inserts the appropriate trace code in the process and then comes back with a message for each process, indicating the trace option selected.

Adding Work Record, Regenerating Dialog, Executing Dialog

The user then leaves CA ADS Trace and enters ADSC, where the work record AT-LINK-RECORD (version 1) is added to the dialog. The dialog is then recompiled and executed.

Viewing the Replay

After the dialog has either executed successfully or terminated abnormally, the user signs back on to CA ADS Trace to view the replay. A typical replay sequence is described later in Typical Sequence (see page 413).

Building Trace Code

The following screen shows the Build Trace Code screen with a dialog name entered. Here are descriptions of the fields.

**DIALOG NAME** -- The user enters the name of the dialog to be traced in the space under DIALOG NAME. In the following figure the user has entered a dialog module named ACDPAY01. After the dialog name is entered, CA ADS Trace responds by listing the processes in the dialog.
**VER** -- The version number of the dialog. The user can enter a version number or use the default. The default is 0001. The second screen shows the Build Trace Code screen after the dialog name has been processed.

**PROCESSES** -- The names of the processes in the dialog.

**TRACE ON** -- The column in which letters indicating the trace options are entered for each process to be traced. The trace options are listed at the bottom of the screen. You can choose one or more of the processes.

```
ADST            Rnn.nn BUILD TRACE CODE hh:mm:ss mm/dd/yy
DICTIONARY: ACCT PAGE:  1 OF:  1
                NAME ON OFF PROCESS NAME VER N MESSAGE
-              ACDPAY01
VER 0001
TRACE ON OPTIONS--
A=TRACE ALL WITH CTL
X=TRACE ALL NO CTL REPLAY LINES LIMIT: 3000
D=DEFINE SUBRTN ONLY LINES AVAILABLE: 2895
E=EXHIBITS ONLY GENERATE WAIT INTERVAL: 75
V=DB VERBS ONLY
I=INTERNAL TRACE
PF1=REVIEW TRACE STATUS
```

**Confirmation**

The following screen shows the Build Trace Code screen after CA ADS Trace has inserted trace code in the processes.

**Confirmation Message** -- The message PROCEED WITH ADSG FOR TRACED DIALOGS - INCLUDE AT-LINK-RECORD appears in the common message area as a reminder to the user to include AT-LINK-RECORD (version 1) on the ADSC work record screen.

**MESSAGES** -- The generator confirms the options selected by displaying a message next to each traced process.

```
ADST            Rnn.nn BUILD TRACE CODE hh:mm:ss mm/dd/yy
DICTIONARY: ATGT PAGE:  1 OF:  1
                NAME ON OFF PROCESS NAME VER N MESSAGE
-            ATGT021I PROCEED WITH ADSG FOR TRACED DIALOGS - INCLUDE AT-LINK-RECORD
ADST            Rnn.nn BUILD TRACE CODE hh:mm:ss mm/dd/yy
DICTIONARY: ACCT PAGE:  1 OF:  1
                NAME ON OFF PROCESS NAME VER N MESSAGE
- ADST ACDPAY01-PM-GET-DATE TRACE OFF
VER 0001 X ACDPAY01-RP-PF15-EXIT TRACE OFF
D ACDPAY01-RP-PF3-END TRACE OFF
A ACDPAY01-RP-PF7-UP TRACE ON
E ACDPAY01-RP-PF8-DOWN TRACE OFF
TRACE ON OPTIONS--
A=TRACE ALL WITH CTL
X=TRACE ALL NO CTL REPLAY LINES LIMIT: 3000
D=DEFINE SUBRTN ONLY LINES AVAILABLE: 2895
E=EXHIBITS ONLY GENERATE WAIT INTERVAL: 75
V=DB VERBS ONLY
I=INTERNAL TRACE
PF1=REVIEW TRACE STATUS
```
Adding Work Record, Recompiling Dialog

The user presses the CLEAR key to exit the CA ADS Trace system, and the System Prompt screen appears. The user proceeds to ADSC, adds the AT-LINK-RECORD (version 1), and recompiles the dialog ACDPAY01.

Executing the Dialog

After the dialog is recompiled, it is executed.

Reviewing Trace Status

The Review Trace Status screen lists dialogs that have been recompiled. The following screen shows an example of processes listed on this screen. By placing the cursor on one of the process names in the list and pressing PF2, the user can transfer the process name and version to the Build Trace Code screen. CA ADS Trace automatically enters the name and version of the dialog.

Trace Replay

Contents
- Typical Sequence (see page 413)
- Selecting a Replay (see page 413)
- Printing the Replay, Moving the Replay to a Queue (see page 413)
- Creating a Replay from a Queue (see page 413)
- Removing a Trace (see page 413)
- Correcting an Error (see page 413)
- Removing the Work Record, Recompiling the Dialog (see page 414)
Selecting Replay Options (see page 414)
Common Fields on Replay Screens (see page 414)
Viewing the Replay (see page 415)
Printing the Replay (see page 415)
Moving the Replay to a Queue (see page 416)
Creating a Replay from the Queue (see page 416)
Deleting a Replay (see page 417)
Preparing Dialog for Normal Use (see page 417)
Removing Trace Code (see page 417)
Correcting an Error (see page 418)
Removing Work Record (see page 418)
Recompiling Dialog (see page 418)

Typical Sequence

The following pages more fully describe a typical sequence of events in a trace replay. This sequence is a sample sequence. Trace Replay (see page 412) gives complete details of every replay option.

Selecting a Replay

From the Main Menu, the user enters option 2 to select the Select Replay screen. On the Select Replay screen, CA ADS Trace displays a list of current replays. By entering a V (View Replay) in the OPT (option) column, the user selects a display of the replay on the terminal.

The user reviews the replay of the trace, scrolling up and down by pages, by number of lines, or to the top or bottom of the replay. CA ADS Trace has added information lines. The replay also shows the values of exhibits and any literals entered by the user.

Printing the Replay, Moving the Replay to a Queue

At the end of the day, the user presses PF11 to access the Print Replay screen and print the replay. Then he/she presses PF10 to access the Move Replay screen and move the replay to a queue. The user then exits CA ADS Trace by pressing the CLEAR key.

Creating a Replay from a Queue

The next day, the user creates the replay from the queue, studies it a while, and, since a printed copy exists, deletes it. CA ADS Trace confirms the delete request before carrying it out. While studying the printed copy of the replay, the user finally finds an error in the process.

Removing a Trace

To remove the trace code from the dialog, the user enters an A in the TRACE OFF column on the Build Trace Code screen. CA ADS Trace removes all trace code created by CA ADS Trace, as well as the exhibit and literal statements entered by the user, and then displays a TRACE OFF message next to the process name.

Correcting an Error

The user exits CA ADS Trace and corrects the error in the source code.
Removing the Work Record, Recompiling the Dialog

Before executing the dialog in the application, the user goes into ADSC, removes the work record, and recompiles the dialog.

Selecting Replay Options

The Select Replay screen is chosen from the Main Menu. It functions as a secondary menu of replay options. On the display is a list of dialogs for which replays exist. This screen offers four choices:

- View Replay
- Move Replay
- Print Replay
- Delete Replay.

On the sample screen the user placed a V in the OPT (option) column to view the replay at the terminal.

<table>
<thead>
<tr>
<th>ADST</th>
<th>Rnn.nn</th>
<th>SELECT REPLAY</th>
<th>hh:mm:ss</th>
<th>mm/dd/yy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICTIONARY: ACCT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONS: V=VIEW REPLAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M=MOVE REPLAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P=PRINT REPLAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=DELETE REPLAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPT</td>
<td>REPLAY NBR</td>
<td>DIALOG</td>
<td>VERSION NBR</td>
<td>NBR OF LINES</td>
</tr>
<tr>
<td>V</td>
<td>1</td>
<td>ACDPAY01</td>
<td>0001</td>
<td>0250</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>ACDPAY01</td>
<td>0001</td>
<td>0043</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>ACDPAY01</td>
<td>0001</td>
<td>0125</td>
</tr>
</tbody>
</table>

Common Fields on Replay Screens

Several fields are common to the replay screens.

**DICTIONARY** -- Name of the dictionary. This field is only for information. To change the dictionary, return to the Main Menu.

**REPLAY NUMBER** -- The number assigned to this replay of the dialog. Every time the dialog is executed with the trace on, a new number is assigned for the replay of each process. If three processes are traced within the dialog, each process is assigned a different number.

Replays moved back from the queue receive new numbers. The total number of replay numbers that can be assigned for one dialog is 100. To generate more replays after this limit is reached, leave CA-IDMS or sign off on the System Prompt screen. This action clears the scratch area.

**DIALOG** -- Name of the dialog.

**VERSION NBR** -- The version number of the dialog.
Viewing the Replay

The following screen shows a page of a sample trace replay. You can move between screens by using PF7 and PF8 for UP and DOWN, or PF5 and PF6 for the first and last pages of the display.

Here are descriptions of the fields.

**LINE nnnn OF nnnn** -- The line number of the line at the top of the display and the total number of lines in the display.

**SKIP** -- The number of lines to skip up or down. Enter a number in this field and press PF7 or PF8 to move up or down a specified number of lines.

**SEARCH** -- An alphanumeric character string to be found. Type in up to 20 characters (including blanks) and press PF7 or PF8 to initiate the search. PF7 initiates an upward search, toward line 1. PF8 initiates a downward search, toward the last line.

**Line Number**

**Trace Descriptor** -- A word that identifies the type of trace content that follows.

**Trace Content** -- Contents of the trace at the current point in execution of the dialog.

**PF Keys** -- Special PF keys assigned to this screen.

**Examples of data lines**

In the typical sequence of events, after viewing the replay, the user presses PF11 to print the replay.

```
ADST     Rnn.nn   VIEW REPLAY   hh:mm:ss mm/dd/yy
DICTIONARY: DEMO             LINE:   16 OF:  19
ATRP007I END OF ADS/O TRACE DATA
REPLAY NUMBER:  2      DIALOG: ICDVPL01      VERSION NBR:  0001
SKIP:         SEARCH:
  16 DBACCESS OBTAIN NEXT PART WITHIN VENDOR-PART.
  17 DBACCESS ERROR-STATUS = 0000
  18 DBACCESS OBTAIN NEXT PART WITHIN VENDOR-PART.
  19 DBACCESS ERROR-STATUS = 0307
```

**Printing the Replay**

The following screen shows the Print Replay screen. The field descriptions are listed below.

**PRINT OPTION** -- Enter P to print the replay. You can then switch directly to the Move Replay screen by pressing PF10, or return to the Select Replay screen (PF3) or to the Main Menu (PF4).

Enter D to both print and delete the replay.

**PRINTER CLASS** -- Enter an integer from 1 through 64 to designate the printer class.

Default value: 1
PRINT DESTINATION -- (optional) Enter a destination for the printer. If you enter a printer destination, do not enter a printer class.

LINES PER PAGE (50 THRU 99) -- Enter the number of lines per page for printing replays.

Default value: 55

In the typical sequence, after printing the replay, the user presses PF10 to access the Move Replay screen.

ADST Rnn.nn PRINT REPLAY hh:mm:ss mm/dd/yy
DICTIONARY: ACCT
REPLAY NUMBER: 3 DIALOG: ACDPAY01 VERSION NBR: 0001
OPTIONS: P=PRINT REPLAY
D=PRINT REPLAY AND DELETE
PRINT OPTION: PRINTER CLASS (1 THRU 64): 1
PRINTER DESTINATION: LINES PER PAGE (50 THRU 99): 55

Moving the Replay to a Queue

The field descriptions listed below correspond to the fields on the Move Replay screen shown below.

REPLAY DESCRIPTION -- A brief description (up to 20 characters) that helps you identify the trace replay. The description is displayed on the Create Replay from Queue screen where this replay is listed.

QUEUE FILE RETENTION PERIOD -- The time period, in days, indicating how long this particular replay is to be retained in the queue file.

Default: 1 day

When these fields have been entered, the Move Replay screen reappears with a message that the replay has been moved to the queue.

When the replay is moved to the queue, it is deleted from the scratch area. The user cannot choose this replay from the Select Replay screen without creating it from the queue.

In the typical sequence, the user presses CLEAR to exit to the System Prompt screen. After lunch, the user comes back to CA ADS Trace and selects option 3, Create Replay from Queue.

ADST Rnn.nn MOVE REPLAY hh:mm:ss mm/dd/yy
DICTIONARY: ACCT
REPLAY NUMBER: 3 DIALOG: ACDPAY01 VERSION NBR: 0001
REPLAY DESCRIPTION: QUEUE FILE RETENTION PERIOD IN DAYS (1 THRU 9): 1

Creating a Replay from the Queue

The following screen shows the CA ADS Trace Create Replay from Queue screen, with the names of all dialogs that have been copied to the replay queue file. If there are more replays listed than will fit on one screen, use the PF keys for scrolling up and down.

In the typical sequence of replay events, the user enters a C in the OPT (option) column. The C instructs CA ADS Trace to send the replay back to the replay selection list on the View Replay screen. When a replay has been created from the queue, it is deleted from the queue.
Deleting a Replay

After viewing the replay for a while, the user deletes it, because a printed copy exists. When PF12 is pressed, the Delete Replay screen is displayed so the deletion can be confirmed.

Preparing Dialog for Normal Use

Before using the traced process in the dialog, you must remove the trace statements, correct the errors in the dialog, remove the work record from the source code, and recompile the dialog.

Removing Trace Code

To remove trace statements, reenter CA ADS Trace and access the Build Trace Code screen. Enter either an A or an X in the TRACE OFF column. If you enter an A, the system removes all of the trace code, including user-entered EXHIBIT and LITERAL statements, and displays a TRACE OFF message next to the process name. The following screen is an example of the Build Trace Code screen with TRACE OFF options entered.

If you enter an X in the TRACE OFF column, CA ADS Trace removes all of the generated trace code, but leaves the exhibits and literals. The only valid characters in the TRACE OFF column are A and X.
Correcting an Error

In this typical sequence, the user finds the error in the process while studying the printed copy of the replay. You can correct an error by leaving CA ADS Trace and modifying the source code with CA IDMS DME or CA IDMS IDD Online.

Removing Work Record

Leave CA ADS Trace, enter ADSC, and remove the work record by spacing over the entry in the WK field on the Work Record screen.

Recompiling Dialog

Recompile the dialog.

Trace Generation Overview

This section contains an overview of trace generation, and an alphabetically organized reference section containing complete descriptions of all of the procedures used in generating or removing the trace.

- Introduction to Trace Generation (see page 419)
- Add Exhibits and Literals (see page 421)
- Access CA ADS Trace (see page 421)
- Build Trace Code (see page 421)
- Add Work Record, Recompile and Execute Dialog (see page 421)
- Review Trace Status of Dialogs (see page 421)
- Multiple Users (see page 422)
- Changing the Trace Option (see page 422)
- Specifying Trace Options (see page 422)
- Trace Entire Dialog (see page 423)
- Trace Processes Within a Dialog (see page 423)
- Trace Included Modules (see page 423)
- Completing Trace Generation (see page 424)
- Change the Trace Option (see page 424)
- Trace Selection Tables (see page 424)
- Trace On Selections (see page 425)
- Trace Off Selections (see page 425)
- Trace Generated Only if Command is First Word (see page 425)
- DATABASE VERBS ONLY (V) (see page 426)
- DEFINE SUBRTN ONLY (D) (see page 426)
- EXHIBITS (see page 427)
- Removing EXHIBIT Statements (see page 427)
- EXHIBITS ONLY (E) (see page 427)
- Generate Wait Interval (see page 427)
Introduction to Trace Generation

The following diagram illustrates the procedure for trace generation.
CA IDMS - 19.0

CA-IDMS/Dictionary Module
Editor or Online DD

Add EXHIBITS
or LITERALS
or Internal Trace

System Prompt

CA-IDMS/ADS TRACE
BUILDING A TRACE

CA-IDMS/ADS TRACE Entry

Main Menu

Build Trace Code

Review Trace Status

System Prompt

ADSC
Add work record
Recompile Dialog

Execute Dialog

Introduction (1)
Add Exhibits and Literals

If element values (exhibits) or literal statements are to be displayed in the trace, add them before accessing CA ADS Trace. You can insert them directly into the source code by using the CA IDMS DME (Dictionary Module Editor). At this time, you can also add TRACE ON and TRACE OFF statements for an internal trace.

Access CA ADS Trace

Then, from the System Prompt screen, enter the task code for CA ADS Trace. At the CA ADS Trace entry screen, press the ENTER key, and the CA ADS Trace Main Menu is displayed. For trace generation, select the Build Trace Code screen from the menu.

Build Trace Code

The Build Trace Code screen lists several options. After the options have been entered for the processes to be traced, CA ADS Trace automatically enters appropriate statements in the source code for the processes, and then displays a message opposite each process name. The message indicates which trace option was applied to the process.

Add Work Record, Recompile and Execute Dialog

After the trace statements have been added to the process source code, you must specify, on the the CA ADS Dialog Compiler (ADSC) Work Record screen, the CA ADS Trace work record (AT-LINK-RECORD, version 1) for the dialog. The dialog can then be recompiled and executed.

Whether the dialog executes successfully or not, a replay of the trace is ready in CA ADS Trace. If the dialog terminated abnormally, the trace continues right up to the termination, with no rollback. If abnormal termination occurs before any trace statements in the dialog, no lines appear under the header in the replay.

Trace Replay (see page 412), describes replay alternatives.

Review Trace Status of Dialogs

To see which dialogs currently contain trace statements, select Review Trace Status, option 4, from the Main Menu.

<table>
<thead>
<tr>
<th>PROCESS NAME</th>
<th>VER</th>
<th>TRACE OPTION</th>
<th>USER ID</th>
<th>DATE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDPAY01-PM-GET-DATE</td>
<td>0001</td>
<td>TRACE ON</td>
<td>TPC12251</td>
<td>mmddyy</td>
<td>hhmm</td>
</tr>
<tr>
<td>ACDPAY01-RP-PF15-EXIT</td>
<td>0001</td>
<td>TRACE ON</td>
<td>TPC12251</td>
<td>mmddyy</td>
<td>hhmm</td>
</tr>
<tr>
<td>ACDPAY01-PR-PF3-END</td>
<td>0001</td>
<td>SUBRTN TRACE ON</td>
<td>TPC12251</td>
<td>mmddyy</td>
<td>hhmm</td>
</tr>
</tbody>
</table>
When the Review Trace Status screen appears, the USER ID field is blank, and all traced processes are listed. To obtain a list of the processes traced by only one user, key in the user id and press ENTER.

From the Review Trace Status screen, you can select one of the processes and pass it to the Build Trace Code screen by positioning the cursor anywhere on the process name and pressing PF2. CA ADS Trace automatically lists the dialog name and version on the Build Trace Code screen.

**Multiple Users**

If another user is going to execute a dialog containing a process that already has trace statements in it, the AT-LINK-RECORD, version 1, must be added on the work record screen in ADSC before the dialog can be recompiled. The second user can check the Review Trace Status screen to see if there are trace statements in any processes being used.

**Changing the Trace Option**

When you want to change a trace option, you do not need to remove the trace before entering the new option on the Build Trace Code screen. For example, if a process was traced using the Define Subroutine option (D), you can then access the Build Trace Code screen and enter an A for Trace All with Control. CA ADS Trace removes the old trace and applies the new one.

**Specifying Trace Options**

Use the Build Trace Code screen to specify the dialog or process names in which tracing is to be turned on or turned off. Enter a dialog name and its version number (default: version 0001), or one or more process names (but not both a dialog name and process names at the same time).

<table>
<thead>
<tr>
<th>ADST</th>
<th>Rnn.nn</th>
<th>BUILD TRACE CODE</th>
<th>hh:mm:ss</th>
<th>mm/dd/yy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICTIONARY: ACCT</td>
<td>DIALOG TRACE NAME</td>
<td>ON</td>
<td>OFF</td>
<td>PROCESS NAME</td>
</tr>
</tbody>
</table>

ACDPAY01 A
VER 0001

TRACE ON OPTIONS--
A=TRACE ALL WITH CTL
X=TRACE ALL NO CTL
D=DEFINE SUBRTN ONLY
E=EXHIBITS ONLY
V=DB VERBS ONLY
I=INTERNAL TRACE

PF1=REVIEW TRACE STATUS
Trace Entire Dialog

If you enter a dialog name plus one of the TRACE ON or TRACE OFF options, the generator automatically applies the option specified to all the processes associated with the dialog name entered. This is a fast method of generating trace statements for the entire dialog.

Trace Processes Within a Dialog

If you key in only a dialog name and then press the ENTER key, CA ADS Trace retrieves all the processes associated with the dialog, and displays the current trace status of each process. You can then enter the trace options to the left of each process name.

Each process may have its own unique TRACE ON or TRACE OFF option. For example, the premap process can have TRACE ALL, option X, turned on, and the response process can have DEFINE SUBRTN ONLY, option D, turned on. Only code that corresponds to the TRACE ON option specified for a process will be generated.

Trace Included Modules

CA ADS Trace does not trace included modules unless you key in the name of the process, indicate an option, and place any character in the IN column. For more information, see INCLUDE (IN) (see page 428).

Note: If trace statements are inserted in processes that are used by other dialogs, those dialogs cannot be recompiled unless you add the AT-LINK-RECORD, version 1, to the work record screen in ADSC. Consider carefully adding trace statements to processes that already execute successfully.
Completing Trace Generation

Once trace statements have been entered in the code, the message TRACE ON or the message TRACE ON/EXHIBIT ON appears, confirming the insertion of trace statements in the code.

Then proceed to ADSC, add the special work record, and recompile the traced dialog. Once the dialog has been successfully recompiled, it is ready for execution as usual. As the dialog executes, CA ADS Trace accumulates trace statistics that are transparent to the application developer. When the dialog has been executed, the CA ADS Trace replay facility is ready to playback the dialog’s execution path.

Change the Trace Option

If you want to change the trace option for a process, access the Build Trace Code screen and enter the new option in the TRACE ON column. CA ADS Trace removes any unnecessary trace statements before inserting the new statements.

<table>
<thead>
<tr>
<th>ADST</th>
<th>Rnn.nn</th>
<th>BUILD TRACE CODE</th>
<th>hh:mm:ss</th>
<th>mm/dd/yy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICTIONARY: ACCT</td>
<td>PAGE: 1 OF: 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIALOG</td>
<td>TRACE</td>
<td>NAME</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>ACDPAY01</td>
<td>A</td>
<td>ACDPAY01-PM-GET-DATE</td>
<td></td>
<td>TRACE OFF</td>
</tr>
<tr>
<td>VER 0001</td>
<td>X</td>
<td>ACDPAY01-RP-PF15-EXIT</td>
<td></td>
<td>TRACE OFF</td>
</tr>
<tr>
<td>D</td>
<td>ACDPAY01-RP-PF3-END</td>
<td>TRACE OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ACDPAY01-RP-PF7-UP</td>
<td>TRACE OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>ACDPAY01-RP-PF8-DOWN</td>
<td>TRACE OFF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TRACE ON OPTIONS--

A=TRACE ALL WITH CTL
X=TRACE ALL NO CTL
D=DEFINE SUBRTN ONLY
E=EXHIBITS ONLY
V=DB VERBS ONLY
I=INTERNAL TRACE

REPLAY LINES LIMIT: 3000
LINES AVAILABLE: 3000
GENERATE WAIT INTERVAL: 75
PF1=REVIEW TRACE STATUS

Trace Selection Tables

The following tables show the selections available on the Build Trace Code screen, with brief descriptions of each alternative. The rest of this section gives complete information on each of these alternatives. They are presented in alphabetical order.
## Trace On Selections

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A TRACE ALL WITH CTL</td>
<td>Trace all control commands, DEFINE SUBROUTINE statements, exhibits, literals, and database commands.</td>
</tr>
<tr>
<td>X TRACE ALL NO CTL</td>
<td>Trace all of the above except control commands.</td>
</tr>
<tr>
<td>D DEFINE SUBRTN ONLY</td>
<td>Trace only DEFINE SUBROUTINE commands.</td>
</tr>
<tr>
<td>E EXHIBITS ONLY</td>
<td>Trace only the exhibit statements and display the values of the elements exhibited.</td>
</tr>
<tr>
<td>V DB VERBS ONLY</td>
<td>Trace only the database verbs and display the error status of each verb.</td>
</tr>
<tr>
<td>I INTERNAL TRACE</td>
<td>Trace the segment of the dialog marked by user-inserted trace statements. This option is equivalent to a TRACE ALL, except the trace applies only to a segment of the process module.</td>
</tr>
<tr>
<td>LINE LIMIT</td>
<td>For each dialog, place a limit on the number of lines to be stored in scratch, so that a loop does not fill up the space. 3000 to 5000. Default: 3000.</td>
</tr>
<tr>
<td>LINES AVAILABLE</td>
<td>Lists the number of lines remaining in the scratch area for this dialog.</td>
</tr>
<tr>
<td>GENERATE WAIT INTERVAL</td>
<td>Allows control of the number of times the interrupt routine is called during module trace generation. 1 to 500. Default: 75.</td>
</tr>
</tbody>
</table>

## Trace Off Selections

<table>
<thead>
<tr>
<th>Selection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A TRACE OFF ALL</td>
<td>Entered in the TRACE OFF column, removes all CA ADS Trace generated statements and all EXHIBIT and LITERAL statements. Regenerate the dialog.</td>
</tr>
<tr>
<td>X TRACE OFF GENERATED STATEMENTS</td>
<td>Entered in the TRACE OFF column, removes CA ADS Trace generated statements. Recompile the dialog.</td>
</tr>
<tr>
<td>I INCLUDE N</td>
<td>An X in the IN column shows that this process is an included module. At least one process in the list must not be an included module.</td>
</tr>
</tbody>
</table>

## Trace Generated Only if Command is First Word

Trace statements are generated for commands only if one of the commands is the first word on a given source statement line. For example, in the statement

```
IF DB-STATUS-OK
```
IF DB-STATUS-OK
   OBTAIN CALC CUSTOMER-RECORD.

trace statements are generated for the OBTAIN command because it is the first word on the line.

But in the statement

IF DB-STATUS-OK
   THEN OBTAIN CALC CUSTOMER-RECORD.

no trace statements are generated because the word THEN precedes the OBTAIN command.

DATABASE VERBS ONLY (V)

Use the DATABASE VERBS ONLY option to trace for only database verbs (and other commands listed below). The CA ADS Trace replay facility echoes the command and follows the command with the error status. See below for a list of commands that are traced.

CONNECT
DELETE QUEUE
DELETE SCRATCH
DISCONNECT
ERASE
FIND
GET
GET DETAIL
GET SCRATCH
MODIFY
OBTAIN
PUT DETAIL
PUT QUEUE
PUT SCRATCH
STORE
ROLLBACK

CA ADS Trace does not trace PUT NEW DETAIL and PUT CURRENT DETAIL commands.

Use this option as a shorthand method of determining which verbs the dialog went through at execution time. If you require further tracing information, select the TRACE ALL A or X option, and recompile the dialog.

DEFINE SUBROUTINE labels, control commands, exhibits, and literals are not traced when using this option.

DEFINE SUBRTN ONLY (D)

Use the DEFINE SUBRTN ONLY option to trace only DEFINE SUBROUTINE statements. The CA ADS Trace replay facility will show the DEFINE SUBROUTINE labels that the dialog went through during program execution.

Since database verbs and control commands are not traced when using this option, you can use it as a quick method of determining what the dialog did at execution time. Then, if you require further tracing information, select the TRACE ALL A or X option, and recompile the dialog.
EXHIBITS

An exhibit allows you to observe the specific value of an element as the dialog is executed. In most cases, exhibits are used with the TRACE ALL (A or X) option. The option EXHIBITS ONLY also traces exhibit statements.

Add EXHIBIT statements to the process code by using CA IDMS DME Here is the syntax:

EXHIBIT element-name

EXHIBIT element-name OF record-name

If you use an EXHIBIT statement as the first statement of a dialog, the statement must end with a period.

CA ADS Trace comments out the EXHIBIT statement and adds code acceptable to ADSC. The syntax of the commented EXHIBIT statement is:

!<> EXHIBIT element-name.

During subsequent trace generations, you do not need to uncomment this statement; CA ADS Trace recognizes this syntax and adds any code necessary to trace the value of the exhibited variable.

If exhibits for numeric data elements that have decimal points are specified, CA ADS Trace rounds up elements that have values of .5 to .9, and rounds down elements that have values of .1 to .4. For example, if you are exhibiting a field called WK-AMOUNT which has a picture of 999.99 and a value of 100.95, the CA ADS Trace replay shows this value as 101.00. Numeric elements which have no decimal points show the actual value.

Removing EXHIBIT Statements

When tracing is no longer desired, remove the EXHIBIT statements from the process source code or choose TRACE OFF option A. Entering an A in the TRACE OFF column directs CA ADS Trace to remove all trace statements, including EXHIBITS and LITERALS.

EXHIBITS ONLY (E)

Use the EXHIBITS ONLY option to trace only EXHIBIT statements. Database verbs, control commands, and DEFINE SUBROUTINE labels are not traced when using this option.

Generate Wait Interval

The GENERATE WAIT INTERVAL field on the Build Trace Code screen allows control of the number of times the interrupt routine is called during module trace generation. CA ADS Trace issues a HICCUP WAIT to remind CA IDMS that CA ADS Trace is still executing. This reminder prevents a runaway task abnormal termination from occurring during CA ADS Trace generation.

The valid range is from 1 to 500.
Default value: 75

This parameter can be useful when tracing modules that have few DEFINE SUBROUTINE, database verb, control command, or EXHIBIT statements (that is, modules which have many MOVE and/or COMPUTE statements). Since CA ADS Trace does not record the MOVE and COMPUTE statements, the operating system needs a reminder that CA ADS Trace is still running properly.

If the RUNAWAY INTERVAL defined in your CA IDMS sysgen is high (for example, 30 seconds), the default value, 75, can be used.

If the RUNAWAY INTERVAL defined in your CA IDMS sysgen is low (for instance, 10 seconds) a smaller CA ADS Trace generate wait interval should be used.

**INCLUDE (IN)**

CA ADS Trace allows tracing to be turned on or off for included modules. Included modules are not traced unless specifically designated. If you want to trace an entire dialog with tracing of an included module, enter the dialog name and version on the Build Trace Code screen. When the list of processes is displayed, enter trace options for the processes listed. Then key in the included process name and enter the appropriate TRACE ON option and any non-blank character in the IN column.

If the process is not designated in the IN column, CA ADS Trace generates extra subroutines for that process. When tracing is turned on for other non-included processes in the dialog, the following error messages occur during recompiling:

DC167062 DUPLICATE SUBROUTINE NAME

DC157008 UNRESOLVED CALL SUBROUTINE AT END OF PROCESS

If one of these messages appears, return to the Build Trace Code screen and enter a non-blank character in the IN column next to the included process name.

**Tracing Only an Included Process**

During development of a process that will be included in several dialogs, you may want to trace only that process, within a dialog. In that case, on the Build Trace Code screen, enter the process name and the trace option and leave the IN column blank. There must be at least one process in the dialog that is not marked in the IN column.

**Included Process in Several Dialogs**

If an included module (for example, a global response) is used in several dialogs, the included TRACE ON option affects all of the dialogs. To execute each dialog using a traced included process, you must add the AT-LINK-RECORD (version 1) in ADSC. The Review Trace Status screen shows which modules have a trace on.
Internal Trace (I)

You can use an internal trace if TRACE ON and TRACE OFF commands are included within the source module itself.

Any DEFINE SUBROUTINE, database verb, control command or EXHIBIT command encountered within the TRACE ON/TRACE OFF boundaries is traced. The internal trace is the same as a TRACE ALL WITH CONTROL (option A) within the boundaries.

Any commands outside the boundary are not traced. If EXHIBIT commands are placed outside the TRACE ON/TRACE OFF boundaries, recompiling errors occur.

CA ADS Trace generates trace statements only for code that falls within the TRACE ON and TRACE OFF boundaries. If no TRACE OFF statement is found following a TRACE ON, CA ADS Trace generates trace statements until the end of the source code is reached.

Adding TRACE ON and TRACE OFF Statements

Add the TRACE ON and TRACE OFF statements by using CA IDMS DME. You can enter a TRACE ON or TRACE OFF statement anywhere in columns 1 through 72, but it must be wholly contained in one source line and be the only statement in the source line.

CA ADS Trace Comments Commands

When CA ADS Trace encounters a TRACE ON statement, it comments the command as follows:

!>> TRACE ON

During subsequent trace generations, you do not need to uncomment this command. CA ADS Trace recognizes this syntax and turns tracing on at this point.

When CA ADS Trace encounters a TRACE OFF statement, it comments the command as follows:

!<< TRACE OFF

Again, you do not need to uncomment this command for subsequent trace generations. CA ADS Trace recognizes this syntax and turns tracing off at this point.

Only Option (I) is Valid

CA ADS Trace only processes the TRACE ON and TRACE OFF statements if I is entered in the TRACE ON column. If TRACE ON or TRACE OFF statements are included in the source code and any other option is chosen, the statements are not commented out. If the user then tries to recompile the dialog using ADSC, recompilation errors occur. At this point, the user must either physically remove the TRACE ON and TRACE OFF statements, or select option I next to the appropriate process name on the Build Trace Code screen.
If I is specified as a TRACE ON option, and if no TRACE ON or TRACE OFF statements are found in the source code, CA ADS Trace returns the following message for the first module in error and stops processing:

\texttt{ATGT057W OPTION I' SPECIFIED UNDER TRACE ON, BUT NO TRACE ON STMT FOUND}

To correct the situation, either include TRACE ON or TRACE OFF statements in the source code, or choose another option.

\section*{Removing an Internal Trace}

If you want to stop the internal trace, you must remove the TRACE ON and TRACE OFF statements from the process source code.

\section*{Line Limit}

The \texttt{LINE LIMIT} field on the Build Trace Code screen allows control of the number of traced lines to be written to the replay file in the scratch area when the traced dialog is executed.

- **Valid range:** 3000 to 5000 lines
- **Default value:** 3000

A maximum of 5000 traced lines is allowed per dialog. Since the amount of scratch area in CA IDMS is limited, the line limit prevents the scratch area from filling up if a traced dialog gets into a loop.

When the line limit for a dialog is exceeded at execution time, all further tracing on that dialog stops, and a message is displayed after the last line of the replay.

\section*{Changing the Line Limit}

To change the line limit for a particular dialog, access the Build Trace Code screen, enter a new \texttt{LINE LIMIT}. Then execute the dialog again, using the revised line limit.

As long as the dialog still has any trace statements, you can only increase the line limit, not decrease it. If you remove traces from all of the processes within the dialog, then any value from 3000 to 5000 is again valid.

\section*{Limit on Number of Replays}

For a specific dialog, 100 replays are allowed. Every time the dialog is executed, the replay of each process traced receives a new replay number. Replays moved back from the queue also receive new numbers.

Deleting a replay or moving a replay to the queue does not change the dialog line availability, nor does it reclaim the space in the scratch area, unless the replay was the only one remaining in the scratch area for that dialog.
Clearing the Scratch Area

After the replay number limit of 100 is reached or the line limit 5000 is reached, to generate more
replays, leave CA IDMS or sign off on the System Prompt screen. This action clears the scratch area.

Lines Available

The LINES AVAILABLE field tells how many lines are left in the scratch area for this dialog.

Literals

In addition to exhibiting values, the user can display LITERALS that are not defined as elements in
map or work records. The syntax of the LITERAL command is as follows:

LITERAL alphanumeric characters

The limit is 61 alphanumeric characters, including blanks.

If you use a LITERAL statement as the first statement of a dialog, the statement must end with a
period.

For example, the command

LITERAL I AM IN SUBROUTINE PUTMAP05

is a valid example of a literal that may be coded in the process. When CA ADS Trace encounters this
statement, it comments the command as follows:

!<>LITERAL I AM IN SUBROUTINE PUTMAP05

For subsequent trace generations, you do not need to uncomment this statement. CA ADS Trace
recognizes this syntax and adds the code necessary to trace the user-defined literals.

When tracing is no longer desired, remove the LITERAL statement from the source code or choose
TRACE OFF option A. Option A directs CA ADS Trace to remove all trace statements, including
generated trace statements and EXHIBITS, LITERALS, and TRACE ON and TRACE OFF statements.

TRACE ALL WITH CONTROL (A)

The TRACE ALL WITH CTL option is the most powerful of the TRACE ON options. If A is entered in the
TRACE ON column, this option directs CA ADS Trace to generate trace statements for all of the
commands shown below.
TRACE ALL NO CONTROL (X)

The TRACE ALL NO CTL option is similar to the TRACE ALL WITH CTL option except that it does not cause generation of trace statements for control commands. When X is specified, only the Database Commands below cause trace statements to be generated.

Database Commands

CONNECT
DEFINE SUBROUTINE
DELETE SCRATCH
DELETE QUEUE
DISCONNECT
ERASE
EXHIBIT element-name
FIND
GET
GET DETAIL
GET SCRATCH
LITERAL alphanumeric characters
MODIFY
OBTAIN
PUT DETAIL
PUT QUEUE
PUT SCRATCH
ROLLBACK
STORE

Control Commands 1

DISPLAY
DISPLAY CONTINUE
EXECUTE NEXT FUNCTION INVOKE
LEAVE
LINK
RETURN
TRANSFER

Trace Generated Only if Command is First Word 1

Trace statements are generated for the Database and Control commands shown only if one of the commands is the first word on a given source statement line.

For example, in the statement

IF DB-STATUS-OK
   OBTAIN CALC CUSTOMER-RECORD.

trace statements are generated for the OBTAIN command because it is the first word on the line.

But in the statement

IF DB-STATUS-OK
   THEN OBTAIN CALC CUSTOMER-RECORD.
no trace statements are generated because the word THEN precedes the OBTAIN command.

**TRACE OFF (A) or (X) Removing a Trace**

Use the TRACE OFF option to turn tracing off in dialogs or processes that have had one of the TRACE ON options applied to them. After the trace statements are no longer needed, return to the Build Trace Code screen and enter either A or X in the TRACE OFF column for each process.

**TRACE OFF X--CA ADS Trace Statements**

Use TRACE OFF option X to remove all CA ADS Trace generated statements for process source.

**TRACE OFF A--CA ADS Trace and User Statements**

Use TRACE OFF option A to remove all CA ADS Trace generated statements, plus all EXHIBIT and LITERAL commands inserted by the user. This option also removes the internal trace statements TRACE ON and TRACE OFF.

When either A or X is entered in the TRACE OFF column, CA ADS Trace deletes all tracing statements that were placed into the source code; the process code appears as it did before any tracing was done.

When tracing has been turned off, the message TRACE OFF appears next to the appropriate process name. The dialog must then be recompiled.

If future tracing is to be done on the dialog, keep the AT-LINK-RECORD on the ADSC work record screen. If no further tracing is desired, before recompiling, delete the AT-LINK-RECORD by spacing over the character in the WK column on the work record screen.

**ADS Trace Replay**

This section contains an overview of the replay functions of CA ADS Trace and complete descriptions of all options. The replay of a trace can be viewed at a terminal, printed, deleted, or moved to a special queue. It can also be deleted from the queue.

- Conceptual View of Replay Options (see page 434)
- Create Replay from the Queue (see page 437)
- Delete Queue Confirmation (see page 438)
- Delete Replay (see page 438)
- Move Replay (see page 438)
- Print Replay (see page 439)
- Select Replay Screen (see page 440)
- View Replay (see page 441)
Conceptual View of Replay Options

After a dialog containing trace code is executed, when you reenter CA ADS Trace, the Main Menu offers two replay options: Select Replay and Create Replay from Queue.

- Select Replay (see page 435)
- Create Replay from Queue (see page 436)
- Screen Descriptions (see page 436)
- Common Fields on Replay Screens (see page 436)

The following figure diagrams these options.
Select Replay

The Select Replay screen lists the replays available, by dialog. There are four choices: View Replay, Move Replay, Print Replay, and Delete Replay.

The View Replay option presents a replay of the trace on the terminal screen. To find information, you can page up and down, search for a character string, or skip a specified number of lines.

If you anticipate recalling the replay data later, move the replay to a special queue. Before moving the replay to the queue, you can enter a 20-character description to indicate the nature of the trace.
You can also save the replay data for later review by printing a hard copy.

When finished with a specific replay, you can delete it.

Create Replay from Queue

If CA ADS Trace has replays in the queue, you can access the Create Replay from Queue screen by selecting option 3 on the Main Menu.

From the Create Replay from Queue screen, you can delete the replay entirely, or create the replay— that is, move it into the scratch area from the queue so that the dialog name, version number, replay number (newly assigned), and total number of lines are listed on the Select Replay screen. From the Select Replay screen, all of the replay selection options are available.

If you specify delete from queue for a dialog on the Create Replay from Queue screen, a Delete Confirmation screen appears; it offers a chance to cancel the deletion before it is actually carried out.

Screen Descriptions

This section describes all of the options on each replay screen. To make this section easy to use for reference, the screens are presented in alphabetical order.

Common Fields on Replay Screens

Several fields are common to the replay screens. The fields listed here correspond to the fields in the following screen.

**DICTIONARY** -- Name of the dictionary. This field is only for information. To change the dictionary, return to the Main Menu.

**REPLAY NUMBER** -- The number assigned to this replay of the dialog. Every time the dialog is executed with the trace on, the replay of each process receives a new number. If three processes are traced within the dialog, each process is assigned a different number.

Replays moved back from the queue also receive new numbers. The total number of replay numbers that can be assigned for one dialog is 100. To generate more replays after this limit is reached, leave CA IDMS or sign off on the System Prompt screen. This action clears the scratch area, so that you can reenter CA ADS Trace and generate more trace replays.

**DIALOG** -- Name of the dialog. If only the process name was entered when the trace code was built, the dialog name on replay screens is blank.

**VERSION NBR** -- Version number of the dialog.

```
ADST                 Rnn.nn     --- MOVE REPLAY -------------------- hh:mm:ss mm/dd/yy
DICTIONARY: ACCT
REPLAY NUMBER: 3     DIALOG: ACDPAY01     VERSION NBR: 0001
REPLAY DESCRIPTION: QUEUE FILE RETENTION PERIOD IN DAYS (1 THRU 9): 1
```
Create Replay from the Queue

Use the CA ADS Trace, Create Replay From Queue screen to review all dialog replays that have been moved to the queue file. The screen appears when you enter option 3 on the Main Menu screen. When the screen is displayed, the USER ID field is blank, and all replays in the queue are listed. To obtain a list of the replays generated by one user, type in the user ID in the USER ID field and press the ENTER key.

- Deletion of a Replay from the Queue (see page 437)

If you enter a C in the OPT (option) column, the replay will be ready for selection from the Select Replay screen. A new replay number is assigned when the replay is created from the queue. You can select only one dialog at a time.

A replay that has been sent to the replay scratch area is deleted from the queue file. Before creating the replay, you may want to compare the number of lines in the replay against the number of lines available listed on the Build Trace Code screen.

If more than one dialog is to be selected, type another C next to the appropriate dialog, and press the ENTER key again.

If you enter a D in the OPT (option) column next to the replay number, that version of the replay will be deleted. The other replays do not change their replay numbers. When D is entered for a replay, another screen is displayed to let you confirm the deletion.

If you have not deleted a replay from the queue by the end of the retention period, the replay is automatically deleted at the end of the retention period listed in the RET column. (You can specify the retention period on the Move Replay screen. The default retention period is one day.)
Delete Queue Confirmation

After you have decided to delete a replay from the queue and entered a D on the Create Replay from Queue screen, a confirmation screen is displayed to let you confirm the deletion.

ADST Rnn.nn --- DELETE QUEUE CONFIRMATION ---- hh:mm:ss mm/dd/yy
DICTIONARY: ACCT
REPLAY NUMBER: 2 DIALOG: ACDPAY01 VERSION NBR: 0001
NUMBER OF LINES: 105 USER ID: TPC12251 DESCRIPTION: EXHIBITS
DATE: mm/dd/yy TIME: hh:mm RETENTION: 1
PRESS ENTER TO DELETE QUEUE
OR PRESS PF3 TO CANCEL DELETE

Delete Replay

To access the Delete Replay screen, type D in the OPT (option) column of the Select Replay screen. You can also access the Delete Replay screen by pressing PF12 from the View Replay screen.

Pressing the ENTER key completes the deletion. This deletion does not change the replay numbers of other replays for this dialog.

When PF3 is pressed, the display returns to the Select Replay screen without deleting the replay.

If you delete a replay, you do not reclaim the scratch area space unless the deleted replay was the last replay remaining in the scratch area for the dialog. The scratch area for a dialog only clears when all of the replays for that dialog have been deleted.

ADST Rnn.nn --- DELETE REPLAY -------------- hh:mm:ss mm/dd/yy
DICTIONARY: ACCT
REPLAY NUMBER: 3 DIALOG: ACDPAY02 VERSION NBR: 0001
PRESS ENTER TO DELETE REPLAY
OR PRESS PF3 TO CANCEL DELETE

Move Replay

To access the Move Replay screen, enter an M in the OPT (option) column, to the left of a dialog name on the Select Replay screen. You can also access the Move Replay screen by pressing PF10 from the View Replay or Print Replay screens.

On the Move Replay screen, the Dictionary name, replay number, dialog name, and version number appear near the top of the screen. These fields are only for information.

The other two fields are for entering variables:

**REPLAY DESCRIPTION** -- User-supplied description of the replay. Before moving the replay, you can enter a replay description of up to 20 characters. This description is displayed on the Create Replay from Queue screen. The description can provide information to distinguish one replay from another for the same dialog.

**QUEUE FILE RETENTION PERIOD** -- Number of days the replay is to be retained in the queue file. You can change the queue file retention period by entering an integer from 1 through 9 in this field.
Default value: 1

When these fields have been entered, the Move Replay screen is displayed with a message that the replay has been moved to the queue.

When the replay is moved to the queue, it is deleted from the scratch area. You cannot then choose the replay from the Select Replay screen without first creating it from the queue.

Print Replay

To access the Print Replay screen, enter a P in the OPT (option) column on the Select Replay screen. You also access the Print Replay screen from the View Replay screen by pressing PF11.

The dictionary name, replay number, dialog name, and version number are displayed near the top of the screen for information.

The variable fields are described below.

**PRINT OPTION** -- Enter a P in this field to print the replay and retain it in the scratch area for further online access.

Enter a D to print the replay and delete it from the list.

The following screen shows an example of a printed replay.

Enter a printer class or a printer destination, but not both.

**PRINTER CLASS** -- Enter an integer from 1 through 64 to designate the printer class.

Default value: 1

**PRINTER DESTINATION** (optional) -- Enter a destination for the printer.

**LINES PER PAGE (50 THRU 99)** -- Enter the number of lines per page for printing replays.

Default value: 55
Select Replay Screen

The Select Replay screen displays a list of processes for which replays exist. This screen offers four choices:

- View Replay
- Move Replay
- Print Replay
- Delete Replay

To select one of the replay options, enter the one-letter code in the OPT (option) field to the left of the Replay number. In screen a V has been entered for Replay Nbr 1.

When the Select Replay file contains more replays than fit on one screen, use PF keys to move the "window" (your terminal screen) up and down the file.

PF5 TOP
PF6 BOTTOM
PF7 UP
PF8 DOWN

If you entered any process names without a dialog name, the DIALOG field for the replay is blank on all replay screens. All such replays are included in the 5000 line limit of the "blank" dialog.
View Replay

Contents

- Control Command Replay Separators (see page 442)

The View Replay screen is displayed below; it is followed by a description of its fields.

ADST  Rnn.nn  ---  VIEW REPLAY  ------------------ hh:mm:ss mm/dd/yy
DICTIONARY: DEMO
LINE: 1 OF: 3
ATRP006I BEGINNING OF ADS/O TRACE DATA
REPLAY NUMBER: 2  DIALOG: ICDVEN01  VERSION NBR: 0001
SEARCH:
1 ICDVEN01 ICDVEN01-RP-INQVEN
2 DBACCESS OBTAIN CALC VENDOR.
3 DBACCESS ERROR-STATUS = 0326
PF10=MOVE REPLAY  PR11=PRINT REPLAY  PF12=DELETE REPLAY

LINE nnnn OF nnnn -- The line number of the line at the top of the display and the total number of lines in the display.

Line number -- The number of the replay line.

Trace Descriptor -- One of the following:

- Name of currently executing program
- EXHIBIT (followed by the element-name)
- VALUE— (followed by the value of the element at this point in dialog execution)
- LITERAL (followed by a user-supplied literal)
- DBACCESS (followed by a database verb or the error status of a preceding verb).

SEARCH -- The Search parameter allows you to scan for a literal string from 1 to 20 characters (including embedded blanks). After entering the string, press PF7 (UP) or PF8 (DOWN). The replay text lines with matching SEARCH characters are highlighted as the first text line on the View Replay screen. To find another occurrence of replay text lines with the same SEARCH characters, press PF7 or PF8 again. While an entry exists in this field, the SKIP value is ignored. To begin scrolling by skip again, delete the characters in the SEARCH field.

SKIP -- The SKIP parameter allows you to control the number of replay text lines to skip in a forward or backward direction. To skip a specified number of lines, enter the number of lines in the SKIP field and press PF7 (UP) or PF8 (DOWN). The valid skip range is from 1 through 5000, but less than the total number of lines in the replay (see LINE). If the number specified is greater than the number of lines to the top or bottom of the display, the first or last page is displayed.

Trace Content -- The content of the trace at that point in execution.

PF Keys -- Special PF keys to switch directly to other replay screens.
Control Command Replay Separators

In order to help the user see more graphically the flow of control of the traced dialog, CA ADS Trace displays separator lines before the control commands. The control commands are traced when A (TRACE ALL--With Control) is specified during trace generation.

Depending on the control command found, one of the following separator lines appears at the point in the dialog where the command is executed:

<table>
<thead>
<tr>
<th>Control Command</th>
<th>Separator Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY</td>
<td>....ABOUT TO DISPLAY....</td>
</tr>
<tr>
<td>DISPLAY CONTINUE</td>
<td>....ABOUT TO DISPLAY CONTINUE....</td>
</tr>
<tr>
<td>EXECUTE NEXT FUNCTION</td>
<td>....ABOUT TO EXECUTE NEXT FUNCTION....</td>
</tr>
<tr>
<td>INVOKE</td>
<td>....ABOUT TO INVOKE DIALOG....</td>
</tr>
<tr>
<td>LEAVE</td>
<td>....ABOUT TO LEAVE....</td>
</tr>
<tr>
<td>LINK</td>
<td>....ABOUT TO LINK....</td>
</tr>
<tr>
<td>RETURN</td>
<td>....ABOUT TO RETURN....</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>....ABOUT TO TRANSFER TO DIALOG....</td>
</tr>
</tbody>
</table>

CA ADS Trace displays the above information only when the control command is the first word on a given source line text.

If the process has only one line of code, the separator line is not displayed.