

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Notices" on page v.

**First Edition (December 1995)**

This edition applies to the licensed program Application Dictionary Services/400 (Feature 2212), Version 3, Release 6, Modification Level 0, a feature of the IBM Application Development ToolSet for OS/400 (Program 5716-PW1) product, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

Because the changes and additions are extensive, this publication should be reviewed in its entirety.

Order publications through your IBM representative or the IBM branch office serving your locality. If you live in the United States, Puerto Rico, or Guam, you can order publications through the IBM Software Manufacturing Solutions at 800+879-2755. Publications are not stocked at the address given below.

A form for reader comments is provided at the back of this publication. If the form has been removed, you can mail your comments to:

IBM Canada Ltd. Laboratory  
Information Development  
2G/345/1150/TOR  
1150 Eglinton Avenue East  
North York, Ontario, Canada. M3C 1H7

You can also send your comments by facsimile (attention: RCF Coordinator), or you can send your comments electronically to IBM. See "Communicating Your Comments to IBM" for a description of the methods. This page immediately precedes the Readers' Comment Form at the back of this publication.

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

© **Copyright International Business Machines Corporation 1994, 1995. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Notices</b> .....	v
Trademarks .....	v
<b>About This Book</b> .....	vii
Who Should Use This Book .....	vii
How This Book is Organized .....	vii
Conventions Used in This Book .....	viii

---

## Integrated Language Environment Concepts .....

1

<b>Chapter 1. Overview of Integrated Language Environment Concepts</b> .....	3
What is the Integrated Language Environment? .....	3
Description of the ILE Program Objects .....	4
ILE Programs .....	4
Service Programs .....	5
Modules .....	5
What Is the Difference Between a Module and a Procedure? .....	5
Creating ILE Program Objects .....	6
Distributing Data Items and Procedures Across ILE Program Objects .....	6
Calling ILE Program Objects .....	7
Activating ILE Program Objects .....	8

---

## Self-Study .....

9

<b>Chapter 2. Getting Started</b> .....	11
Organization of the Modules .....	11
The Setting .....	11
A Description of the Sample Application .....	12
The Functionality of the Sample Application .....	12
The Files Used in the Sample Application .....	13
The Programs Used in the Sample Application .....	13
The Logic Flow of the Sample Application .....	14
Installing the Self-Study Libraries and Creating the User Profiles .....	16
<b>Chapter 3. The Learning Modules</b> .....	17
Module 1: Create a New Dictionary to Contain the Program Modules .....	17
What is an AppDict Services/400 Dictionary? .....	17
Steps to Create the Dictionary .....	18
Checking the Status of the Dictionary Being Created .....	19
What You Have Learned in Module 1 .....	20
Module 1 Questions .....	20
Module 2: Familiarize Yourself with the Structure of the Sample Application .....	21
Familiarize Yourself with the Bound Objects .....	21
Familiarize Yourself with the Binding Programs .....	22
What You Have Learned in Module 2 .....	23
Module 2 Questions .....	23
Module 3: Check the Current Field Length and Type .....	25
Locate the MLZIP Field .....	25
Look at the Record Format Which Contains the Field Description .....	26

Locate the Field and Display Its Length and Type . . . . .	28
What You Have Learned in Module 3 . . . . .	29
Module 3 Questions . . . . .	29
Module 4: Assess the Effect of the Design Change . . . . .	30
Determine Which Fields Would Be Affected . . . . .	30
Determine Which Files Would Be Affected . . . . .	33
Determine Which Programs or ILE Objects Would Be Affected . . . . .	35
What You Have Learned in Module 4 . . . . .	36
Module 4 Questions . . . . .	36
Module 5: Update the Current Field Reference File for Localization . . . . .	38
Edit the MLGREFP File to Change the Length and Type of the MLZIP Field . . . . .	38
What You Have Learned in Module 5 . . . . .	39
Module 5 Questions . . . . .	39
Module 6: Mark MLGREFP as a Field Reference File . . . . .	40
Mark the Field Reference Files . . . . .	40
What You Have Learned in Module 6 . . . . .	40
Module 6 Questions . . . . .	41
Module 7: Search RPG Source for MLZIP Fields Not in Field Reference File . . . . .	42
Search the RPG Source Files . . . . .	42
What You Have Learned in Module 7 . . . . .	43
Module 7 Questions . . . . .	44
Module 8: Recreate All of the Changed Program Objects . . . . .	45
Recreate the objects Affected By the Modified MLGREFP File . . . . .	45
What You Have Learned in Module 8 . . . . .	46
Module 8 Questions . . . . .	46
Module 9: Run the Updated Sample Program . . . . .	48
Run the Program . . . . .	48
What You Have Learned in Module 9 . . . . .	48
Module 9 Questions . . . . .	48
 <b>Chapter 4. Cleaning up the System . . . . .</b>	 49

---

**Answers . . . . . 51**

<b>Appendix A. Answers to the Module Questions . . . . .</b>	53
Module 1 Answers . . . . .	53
Module 2 Answers . . . . .	53
Module 3 Answers . . . . .	53
Module 4 Answers . . . . .	54
Module 5 Answers . . . . .	54
Module 6 Answers . . . . .	54
Module 7 Answers . . . . .	55
Module 8 Answers . . . . .	55
Module 9 Answers . . . . .	55
 <b>Bibliography . . . . .</b>	 57
 <b>Index . . . . .</b>	 59

---

## Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Canada Ltd., Department 071, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, Canada. Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

This publication contains small programs that are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

400	IBMLink
Application System/400	Operating System/400
AS/400	OS/400
C/400	PROFS
COBOL/400	RPG/400
IBM	



---

## About This Book

This book gives you hands-on experience using each of the main functions in Application Dictionary Services/400 (AppDict Services/400), a feature of the Application Development ToolSet/400 product. This book illustrates how to use the AppDict Services/400 features by leading you through a series of step-by-step exercises that are integrated within a larger application development scenario. This scenario illustrates how you can use AppDict Services/400 to help you do your application development work.

For information about other AS/400 publications, see either of the following:

- The *Publications Reference* book, SC41-4003, in the AS/400 Softcopy Library.
- The *AS/400 Information Directory*, a unique, multimedia interface to a searchable database containing descriptions of titles available to IBM or from selected other publishers. The *AS/400 Information Directory* is shipped with your system at no extra charge.

Refer to the “Bibliography” on page 57 for other books you may need for reference.

---

## Who Should Use This Book

This book is intended for application programmers or analysts who want to familiarize themselves with the AppDict Services/400 feature in order to do impact analysis, examine application calling hierarchies, or any of the other AppDict Services/400 functions, which provide information about program parts and their relationship to other program parts.

You should have knowledge of the AS/400 system and commands, and be familiar with programming in the AS/400 environment.

Note that the book is intended to be only a quick hands-on introduction to the main AppDict Services/400 functions. For more detailed information, see the *ADTS/400: Application Dictionary Services/400 User's Guide*, SC09-2087.

If you are interested in IBM customer education, contact your IBM marketing representative for information regarding customer education classes being held at IBM education centers.

---

## How This Book is Organized

This self-study guide contains:

- An overview of the Integrated Language Environment (ILE) concepts
- Background information on the organization of the modules, the setting for the tasks outlined in the self-study, the sample program used in the self-study, and how to set up the system for the self-study
- Self-study modules that focus on the main functions of AppDict Services/400
- Instructions on how to clean up the system after the modules have been completed
- An appendix that provides the answers to the questions in the modules

---

## Conventions Used in This Book

Convention	Meaning
ALL CAPS	Commands, keywords, and names <sup>1</sup>
<i>Italics</i>	<ul style="list-style-type: none"><li>• Titles of manuals</li><li>• Prompts on displays</li></ul>
Monospace type	Information that you type on a display <sup>1</sup>

**Note:**

1. Although this book always shows commands and entries in uppercase letters, you can use uppercase, lowercase, or mixed-case letters.

---

---

# Integrated Language Environment Concepts



---

## Chapter 1. Overview of Integrated Language Environment Concepts

In order to introduce you to, or refresh your understanding of, ILE, this chapter briefly discusses the concepts that are a part of ILE, and compares them to the concepts that are a part of the **Original Program Model (OPM)** (the traditional set of functions, processes, and rules provided by the OS/400 operating system to create and run a program).

Only the ILE concepts that are required for you to understand the changes that have been made to Application Dictionary Services/400 (AppDict Services/400) and how AppDict Services/400 now works will be discussed in this chapter. For more detailed information about ILE concepts and their relationships to OPM, refer to:

- *ILE Concepts*
- *ILE Application Development Example*

This chapter discusses ILE under the following topics:

- What ILE is
- Description of the ILE program objects
- Creating ILE program objects
- Distributing data items and procedures across ILE program objects
- Calling ILE program objects
- Activating ILE program objects

---

### What is the Integrated Language Environment?

In previous versions of Application Dictionary Services/400, the **Original Program Model (OPM)** was the only program model supported. The OPM is the traditional set of functions, processes, and rules provided by the Operating System/400 (OS/400) operating system to create and run a program. This version of AppDict Services/400 supports both OPM and ILE programs.

The **Integrated Language Environment (ILE)** is designed to enhance program development and maintenance on the AS/400 by allowing you to create programs that:

- Are modular
- Use multiple programming languages
- Reuse components from other programs, rather than duplicating them across programs
- Have better call performance
- Can be optimized to run faster

In addition to the benefits just described, converting your OPM programs to ILE creates a foundation for the object-oriented environments of the future.

Integrated Language Environment (ILE) support is new to Version 3 Release 6 of AppDict Services/400.

---

## Description of the ILE Program Objects

In ILE, a few additional program units have been introduced. Programmers can still use OPM programs, but they can combine these programs with other program units that are unique to the ILE environment:

- ILE programs
- Service programs
- Modules

In ILE, all of the program parts that are going to be used by a larger application are created separately, and then they are bound together by the CRTPGM (Create Program) or CRTSRVPGM (Create Service Program) commands. The ILE environment does not allow the program parts to be run as individual units, even if their logic is self-contained. Therefore, modules cannot be run individually. However, service programs also cannot be run individually, since they contain a collection of separate procedures and data items.

**Note:** AppDict Services/400 does not support the creation of programs using the CRTBNDxxx command. It is recommended that if you have any single-module programs, you create them with the appropriate CRTxxxMOD command, followed by the CRTPGM command.

## ILE Programs

An **ILE program** is a set of program units that have been bound together to create an executable program. An ILE program shares the following characteristics with an OPM program:

- The program gets control through a **dynamic program call**, which initiates and activates the called program at runtime
- The entry point to the program is the **program entry procedure (PEP)**, which is the compiler-generated code created for a dynamic program call
- The program is identified to the system by the symbol \*PGM (PGM, if you are using AppDict Services/400 from within an Application Development Manager/400 session)

An ILE program has the following characteristics that an OPM program does not have:

- An ILE program is created from one or more module objects. For more information about modules, refer to “Modules” on page 5.
- One or more of the copied modules can contain a PEP.
- You have control over which module contains the PEP.

Also, procedures and data items *cannot* be exported from an ILE program, but procedures or data items *can* be imported from other ILE objects. An ILE program *can* be run as an individual unit.

## Service Programs

A **service program** is a collection of procedures, which can be separately called when bound to another ILE program or service program, and available data items. A service program is represented to the system by the symbol \*SRVPGM (SRVPGM, if you are using AppDict Services/400 from within an Application Development Manager/400 session). Because of the overhead required the first time a service program is called, service programs should contain common services that many other ILE program objects may need. For example, if a date conversion routine is used by many programs within a larger application, it should be placed in a service program; if it is only used in a couple of programs, it should be placed in a module that can be bound to the applications that need it.

The **public interface** to a service program consists of the names of the exported procedures and data items that other ILE objects need to access. The list of procedures and data items that can be exported is defined using **binder language**. A service program can be updated without having to re-create other ILE programs or service programs as long as the export list remains intact or new procedure names or data items are added to the end of the export list.

Service programs cannot be dynamically called, but can be **called by reference**. Calls to service programs are not made using a dynamic call. Instead, programs that call specific service programs are identified as a parameter of the CRTSRVPGM (Create Service Program) command. The system performs some binding to the service program at bind time, but does not copy any runnable code from the service program into the calling program as would be done for a dynamic program call.

## Modules

A **module** is the unit created by an ILE compiler, but it cannot be run. (An OPM compiler produces a program that can be run.) A module is the basic building block for creating ILE programs, and a module is represented to the system by the symbol \*MODULE (MODULE, if you are using AppDict Services/400 from within an Application Development Manager/400 session). A module can consist of one or more procedures and data item specifications, and these procedures and data items can be directly accessed by other ILE objects. Modules can also export and import procedures and data items.

## What Is the Difference Between a Module and a Procedure?

The terms module and **procedure** are virtually synonymous from an RPG and a CL perspective. One source member in CL or RPG creates one module, which is called a procedure when it runs. Other languages may support modules that have multiple procedures. Therefore, the terminology used in ILE is:

- You create or change modules.
- You call or run procedures.

---

## Creating ILE Program Objects

In OPM, a program is created in a single step. The compiler generates the program object which contains additional code. This additional code initializes program variables and provides any necessary code for special processing that is required by the particular language. For example, special processing could include handling any input parameters expected by the program. When the program is started, or called, the additional compiler-generated code becomes the starting point (entry point) for the program.

The process of creating ILE programs involves two steps:

- Compiling the source code into modules.
- Binding the modules into an ILE program or service program.

Binding occurs when the Create Program (CRTPGM) or Create Service Program (CRTSRVPGM) command is run.

The CRTPGM and CRTSRVPGM commands bind:

- Service programs and modules by copy when they are specified in the MODULE keyword
- Service programs by reference (to other service programs or ILE programs) when they are specified in the BNDSRVPGM keyword



### Tip:

If you use a Binding Directory (BNDDIR) with the CRTPGM command, the modules are bound by copy only if they provide exports for unresolved imports. For more information about imports and exports, refer to “Distributing Data Items and Procedures Across ILE Program Objects.”

---

---

## Distributing Data Items and Procedures Across ILE Program Objects

In OPM, support for sharing data between programs in a larger application is limited. Typically, to share data between programs in an application, data has to be passed as parameters to a CALL statement. This method is normally quite effective, except in those cases where the data is not processed by the next program in sequence.

For example, suppose that program A originates a piece of data that is to be processed by program D. If the normal sequence of events is for A to call B, which calls C, which calls D, the parameter has to be passed from program to program, even though neither B nor C uses it.

When you create an ILE program or service program object, you need to specify how other programs can access that program. On the CRTPGM command, you do so with the Entry Module (ENTMOD) parameter. On the CRTSRVPGM command, you do so with the Export (EXPORT) parameter.

When you create a program, service program, or module in AppDict Services/400, the appropriate system display appears for the create command that AppDict

Services/400 is using. These system displays allow you to enter other parameters, such as ENTMOD.

The ENTMOD parameter specifies the module that contains the PEP and **User Entry Procedure (UEP)**. The UEP is the procedure (written by the programmer) that gets control when a dynamic program call is made; in contrast, the PEP is the compiler-written code that is the entry point to an ILE program when a dynamic program call is made.

The EXPORT parameter (along with the Source File (SRCFILE) and Source Member (SRCMBR) parameters) identifies the **public interface** to the service program being created. The public interface defines the data items and procedures that a service program makes available (exports) to other ILE programs or service programs.

---

## Calling ILE Program Objects

Under OPM, programs were written and compiled. The only mechanism of linking programs to other programs was the program call. The program call that OPM programs use exclusively is the dynamic program call. OPM programs can call programs written in the same language as the calling program or a different language. They cannot, however, combine more than one language in a single program.

In ILE, you can call either a program or a procedure, but you must indicate whether the program object you are calling is a program or a procedure. To do this, you use a separate call statement to represent each type of call. Programs are called dynamically and procedures are called statically.

A **dynamic program call** transfers control to either an ILE program object or an OPM program object. Restrictions apply to which ILE program objects can make valid dynamic program calls. OPM programs, ILE programs, and service programs can make dynamic program calls to other OPM programs or ILE programs only. Service programs cannot be dynamically called.

A **static procedure call** transfers control to an ILE procedure. Static procedure calls can be coded in ILE languages only. A static procedure call can be used to call any of the following:

- A procedure within the same module
- A procedure within a separate module within the same ILE program or service program
- A procedure that has been exported from an ILE service program

**Note:** ILE procedure calls are not supported by AppDict Services/400.

---

## Activating ILE Program Objects

The **activation** of a program is the process used to prepare a program to run. Both OPM programs, and ILE programs and service programs must be activated before they can be used within an OS/400 job.

In OPM, programs are activated through a dynamic program call at runtime. When the call is made, the operating system (OS/400) checks to see if the program is activated. If it is, that copy of the program is given control. If it is not, a copy is made, it is then activated, and control is passed to the program copy.

In ILE, programs are activated in two major steps:

1. Allocation and initialization of storage for the program.
2. Completion of the binding of the program to any referenced service programs.

An ILE program activation is created as part of a dynamic program call. ILE manages the process of program activation by keeping track of program activations within an **activation group**. An activation group is the substructure that contains the resources necessary to run programs. ILE uses the activation group that was specified in the Activation Group (ACTGRP) parameter of the CRTPGM or CRTSRVPGM command.

Several activation steps are unique for service programs. Service program activation:

- Starts indirectly as part of a dynamic program call to an ILE program.
- Includes completion of interprogram binding linkages by mapping the symbolic links to physical links.
- Includes signature checking.

An ILE program activated for the first time within an activation group, is checked for binding to any ILE service programs. If service programs have been bound to the program being activated, they are also activated as part of the same dynamic call processing. This process is repeated until all necessary service programs are activated.

Conceptually, the process of activating an OPM program, ILE program, or service program can be viewed as the completion of the binding process started when the programs and service programs were created. The CRTPGM or CRTSRVPGM command saves the name and library of each referenced service program. An index into a table of exported procedures and data items is also saved in the client program or service program at program creation time. The process of service program activation completes the binding step by changing these symbolic references into addresses that can be used at runtime.

After a service program is activated, static procedure calls and static data item references that are made to modules within different service programs are processed. The amount of processing required is the same as would be required if the modules had been bound by copy into the same program. However, modules bound by copy require less activation time than service programs.





---

## Chapter 2. Getting Started

This section provides background information about the purpose of the self-study and instructions on how to set up your AS/400 system for the self-study modules. It describes:

- The organization of the self-study modules
- The setting for the tasks outlined in the self-study
- The sample application used in the self-study
- How to install the self-study libraries and create the user profiles

### Organization of the Modules

The modules in this self-study are organized as follows:

1. You are introduced to what you will complete in the self-study.
2. You are introduced to any concepts that are relevant to the self-study module.

For example, “Module 1: Create a New Dictionary to Contain the Program Modules” on page 17, describes the concept of a dictionary.

3. You are walked through the steps required to complete the module task.
4. You are provided with a summary of the topics covered in the self-study module.
5. You are provided with questions to help you understand and remember what you have learned.

You should be able to complete the self-study in an hour. However, the time required to complete the self-study may vary depending on the response time of your AS/400 system.

Because the self-study has been designed to walk you through a complete development task using AppDict Services/400, each self-study module builds on the previously completed self-study modules. Therefore, you will not be able to complete some of the self-study modules out of sequence and get the same results. But, if you prefer, you can review the tasks and questions without using the system, and still understand how to use AppDict Services/400.

### The Setting

As an application is developed, feedback is usually received from many different sources, and this feedback can cause the initial design concepts of an application to change. This self-study contains nine modules that walk you through a scenario that requires you to assess the impact of a proposed design change to a sample application, Mailing List, and then make that design change.

The design change will require that you change the size and type of the field that holds the zip code in order to accommodate zip codes from other countries. The process of preparing an application for use in other countries is referred to as **localization**.

For example, the Mailing List application was written to accommodate a numeric zip code with 5 characters. However, the Canadian zip code (or postal code, as it is called) has 6 characters, and uses both numbers and letters; and the Mexican zip

code uses 5 numeric characters. Therefore, to localize this application for North America, the zip code field would have to be increased from 5 to 6 characters, and the field type would have to be changed from numeric to alphanumeric.

Because the MLZIP field is used in many different places in the Mail List application, it is defined in a **field reference file**. This is a file that contains definitions of fields that can be referred to by other parts of the application, instead of being defined each time they are used.

This self-study will show you how to assess the impact of changing a field in the field reference file, and then how to modify the size and type of the zip code field in the existing field reference file so that the application can be used in North America, instead of just the United States of America.

To localize the Mailing List application so that it can be used anywhere in North America, this self-study shows you how to:

- Create a dictionary for the sample application files
- Familiarize yourself with the program by viewing the bound objects and binding programs
- Check the current length and type of the field to validate the design change request
- View the affected fields, files, and ILE objects to assess the impact of the design change
- Change the size and type of the MLZIP field in the existing field reference file
- Mark the existing field reference file so that Application Dictionary Services/400 (AppDict Services/400) will recognize it as a field reference file
- Search the RPG source for any MLZIP files that are not referenced from the field reference file, so that they can also be changed
- Recreate and recompile the affected objects
- Run the updated sample application

Before you can do the self-study, you need to install the exercise libraries and set up the user profiles as described in "Installing the Self-Study Libraries and Creating the User Profiles" on page 16.

## A Description of the Sample Application

This section describes the sample application, including:

- Its functionality
- The files it uses
- The programs it uses
- Its logic flow

### The Functionality of the Sample Application

The sample application, Mailing List, allows users to:

- Search for customers based on their account number.
- Maintain the mailing lists in the database.
- Print a listing of customers by account number.

- Create a special analysis report.
- Use any of the Query utilities (accessed by the STRQRY command) on the mailing list.

The sample application contains ten files, three programs, nine modules, and two service programs.

## The Files Used in the Sample Application

Table 1 lists the files that are used in the sample application.

*Table 1. List of the Files Used in the Sample Application*

File	Attribute	Text
MLGINQD	DSPF	Mailing list inquiry display
MLGMNUD	DSPF	Mailing list menu
MLGMSTL	LF	Mailing list label printing logical file
MLGMSTL2	LF	Mailing list keyed by city, state
MLGMSTL3	LF	General-purpose Logical File for querying MLGMSTP
MLGMSTP	PF-DTA	Mailing list master file
MLGMTND	DSPF	Mailing list maintenance display file
MLGNAMD	DSPF	Mailing list name search display file
MLGNAML	LF	Mailing list logical keyed by name.
MLGREFP	PF-DTA	Mailing list field reference file

## The Programs Used in the Sample Application

Table 2 lists the programs, service programs, and modules used in the sample application. Table 3 on page 14 shows how these programs, service programs, and modules are bound.

*Table 2 (Page 1 of 2). List of the Programs, Service Programs, and Modules Used in the Sample Application*

Program	Type	Attribute	Text
MAILLIST	*PGM	CLLE	Mailing list main program
MLGRPTC	*PGM	CLLE	Print one-liner with MLGMSTL2 LF
MLGRPTC2	*PGM	CLLE	General purpose query of MLGMSTP
MLGMTNS	*SRVPGM		Contains modules MLGMTNR, MLGMTNC, and MLGNAMR.
MLGRPTS	*SRVPGM	RPGLE	Contains module MLGRPTR.
MLGINQR	*MODULE	RPGLE	Mailing list inquiry
MLGLBLR	*MODULE	RPGLE	Mailing list label printing
MLGMNUC	*MODULE	CLLE	Mailing list menu program
MLGMTNC	*MODULE	CLLE	Mailing list maintenance functions
MLGMTNR	*MODULE	RPGLE	Mailing list master maintenance functions

Table 2 (Page 2 of 2). List of the Programs, Service Programs, and Modules Used in the Sample Application

Program	Type	Attribute	Text
MLGNAMR	*MODULE	RPGLE	Mailing list name search
MLGRPTC	*MODULE	CLLE	Print one-liner with Logical File MLGMSTL2
MLGRPTC2	*MODULE	CLLE	General-purpose query of MLGMSTP
MLGRPTR	*MODULE	RPGLE	Mailing list one-line report per name

Table 3. How Programs, Service Programs, and Modules Used in the Sample Application Are Bound

Bound Program	Bound Type	Program	Type
MLGMTNS	*SRVPGM	MLGMTNR	*MODULE
		MLGMTNC	*MODULE
		MLGNAMR	*MODULE
MLGRPTS	*SRVPGM	MLGRPTR	*MODULE
MAILLIST	*PGM	MLGMNUC	*MODULE
		MLGINQR	*MDOULE
		MLGMTNS	*SRVPGM
MLGRPTC	*PGM	MLGRPTC	*MODULE
		MLGRPTS	*SRVPGM
MLGRPTC2	*PGM	MLGRPTC2	*MODULE
		MLGRPTS	*SRVPGM

**Note:** Some other ILE run-time service programs that are linked to the sample application programs and service programs will appear in the sample dictionary (such as QLECW, QLEAW, QRNXIE, and QRNXIO). These service programs are present because they are automatically bound to the sample application programs and service programs by the ILE binder.

### The Logic Flow of the Sample Application

Figure 1 on page 15 shows the logic flow of the sample application. When the sample application is started, the Mailing List main menu appears. The main menu allows the user to select from options 1 through 5, which provide access to the Mailing List functions. If the user chooses to exit from the main menu, the Mailing List application is ended.

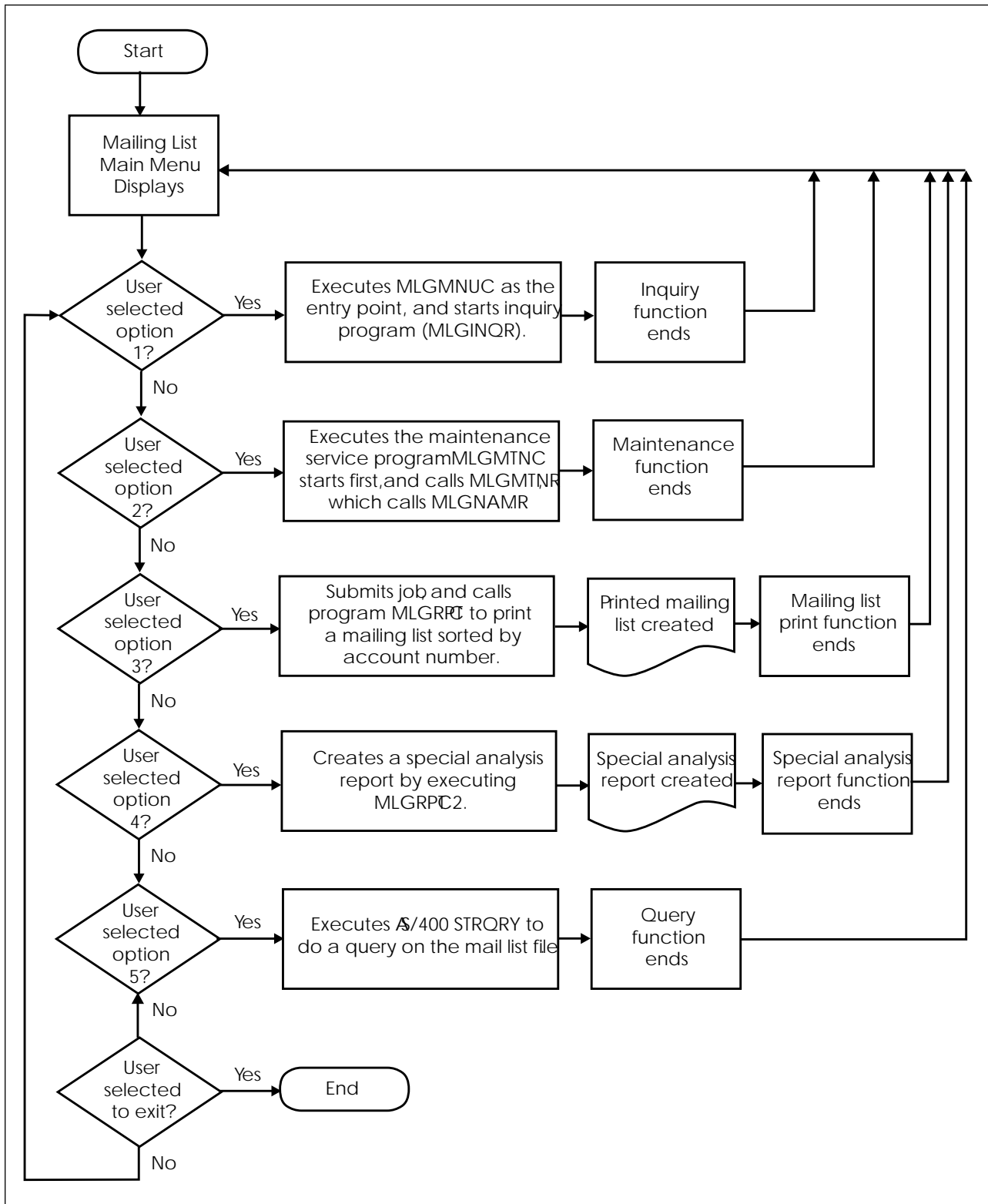


Figure 1. The Flowchart of the Sample Application

---

## Installing the Self-Study Libraries and Creating the User Profiles

To install the Application Dictionary Services/400 libraries required for the exercises in the self-study modules in this book, the system administrator (or anyone else with appropriate authority) should do the following:

1. Determine the maximum number of people at the site who will be doing the self-study.
2. Ensure that the AppDict Services/400 feature is installed on the Application System/400 (AS/400) system.
3. Sign on to the AS/400 system with a user ID that has either:
  - A user class of \*PGMR and special authority of \*ALLOBJ, \*SAVSYS, \*JOBCTL, and \*SECADM
  - A user class of \*SECOFR and default authorities

4. On any command line, enter:

```
RSTLIB SAVLIB(QDMTLAB) DEV(*SAVF) SAVF(QDMT/QDMTLAB)
```

A message appears indicating that the objects have been restored.

5. On any command line, enter:

```
RSTLIB SAVLIB(ADSSRCLIB) DEV(*SAVF) SAVF(QDMTLAB/ADSSRCLIB)
```

A message appears indicating that the objects have been restored.

6. On any command line, enter:

```
ADDLIBLE QDMTLAB
```

A library called QDMTLAB is added to the library list.

7. On any command line, enter the following, where *user-number* is the number of people that are going to be doing the self-study in this session:

```
CALL SETUP PARM(user-number)
```

The sample application parts are created, and a library and user profile are created for the number of users you specified. Each library will be called ADSILEU*nn* and each user profile will be called ADS*nn*, where *nn* represents the number of users you specified. The self-study users create the dictionaries as described in “Module 1: Create a New Dictionary to Contain the Program Modules” on page 17.

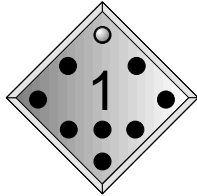
---

## Chapter 3. The Learning Modules

This chapter contains all of the lessons that walk you through the steps required in Application Dictionary Services/400 (AppDict Services/400) to complete the tasks outlined in “The Setting” on page 11.

---

### Module 1: Create a New Dictionary to Contain the Program Modules



The sample application is restored to a library called ADSILEU01. In AppDict Services/400, dictionaries hold your application libraries and the objects contained in them.

This module introduces you to the concept of an application dictionary and how AppDict Services/400 uses a dictionary to store information about the objects used in your applications and the relationships these objects have to each other.

In this module, you also learn how to create a new library to hold the ADSILEU01 library, turn the new library into an AppDict Services/400 dictionary, and how to monitor the status of the dictionary-creation job.

#### Notes:

1. Throughout the rest of this self-study, the dictionary ADSILEDI01 will be used for illustration purposes. For the examples and displays that follow, you should substitute this dictionary name with the actual dictionary that you are working with.
2. Because the sample application files are being added to AppDict Services/400 for the first time, the field reference file needs to be marked as such within AppDict Services/400. We will not do this step in this module. “Module 5: Update the Current Field Reference File for Localization” on page 38 shows you how to mark field reference files.

### What is an AppDict Services/400 Dictionary?

An AppDict Services/400 **dictionary** holds information that describes application objects and indicates their relationships to other application objects. As you make changes to the documented objects in the sample Mailing List application, the AppDict Services/400 dictionary also updates its information about these objects. For more technical information about dictionaries, refer to the *Application Dictionary Services/400 User's Guide*.

**Note:** Dictionary names can be up to 10 characters. They must begin with an alphabetic character (A through Z, a through z, \$, #, or @), and can be followed by up to 9 alphanumeric characters (A through Z, a through z, 0 through 9, \$, #, @, period (.), or underline (\_)). Blanks and quotation marks are not allowed. For example, you could not use SLF STUDY, but you could use SLF\_STDY.

## Steps to Create the Dictionary

To create the dictionary to hold the sample application files:

1. To create the library from which the dictionary will be created, on the command line type the following, and press the Enter key (where *nn* is the user number you are):

```
CRTLIB ADSILEDInn
```

2. To turn the library you just created into a dictionary, do the following:

- a. On the command line type the following, and press the Enter key:

```
WRKAPPDCT
```

The Work with Dictionaries display appears.

- b. Press F6 (Create).

The Create Dictionary display appears.

- c. In the *Application dictionary* prompt, type *ADSILEDI*nn** (which is the library you created in step 1), and press the Enter key.

The Document Libraries for Dictionary display appears.

- d. In the *Opt* prompt of the Document Libraries for Dictionary display type 1 (Add).

- e. In the *Library* prompt, type the name of the library to be documented, and press the Enter key:

```
ADSILEUnn
```

Your display should appear similar to the one shown in Figure 2.

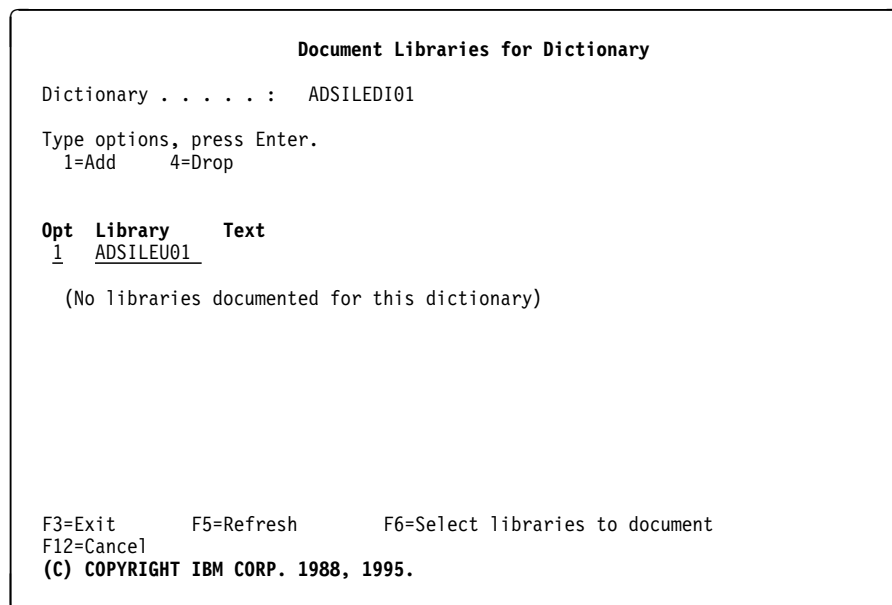


Figure 2. Document Libraries for Dictionary Display, Showing ADSILEU01 Library to Be Added

- f. To submit the dictionary creation job to the batch system, press the Enter key again.

The Work with Dictionaries display appears showing the name of the new dictionary and a message indicating that the dictionary-creation job has been submitted.

You will have to wait until the dictionary has been created before you can use it. The rest of this section and the next section, "Checking the Status of the Dictionary Being Created," describes how to find out when a dictionary has finished being created.

The dictionary-creation takes some time to complete, so the new dictionary will not be available to use immediately.

If the ADSILEDInn dictionary is still being created, its status will be Processing. When the status has changed to Not in use, you can begin to work with the dictionary.

Figure 3 shows the ADSILEDI01 dictionary with a status of Processing.

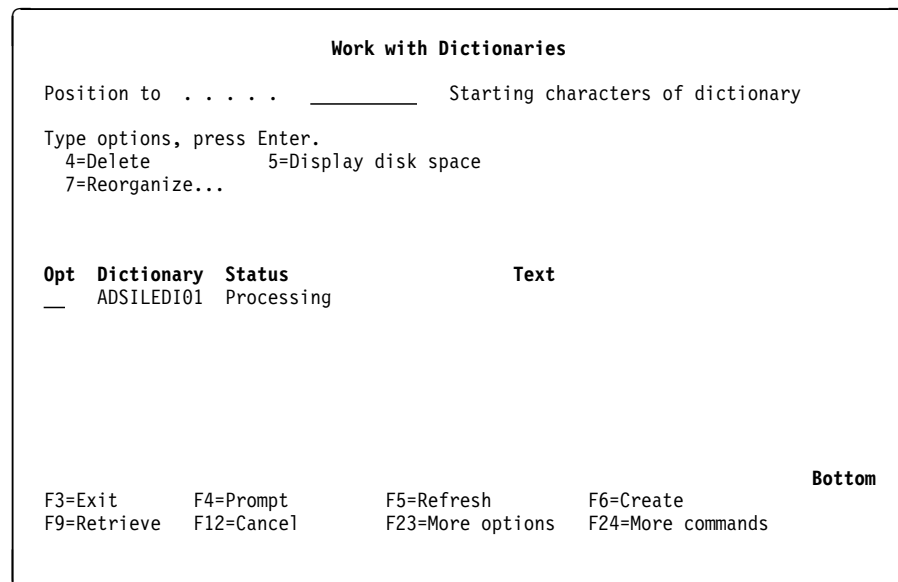


Figure 3. Work with Dictionaries Display, Showing ADSILEDI01 Dictionary Processing

- g. Press F5 (Refresh) until the status changes from Processing to Not in use.

The dictionary is now ready to use. The next section describes another method that you can use to check the status of a dictionary being created.

- h. Press F12 (Cancel).

The AS/400 Main Menu appears.

### Checking the Status of the Dictionary Being Created

The other method you can use to determine when a dictionary has finished being created is to use the AS/400 DSPMSG command.

1. To use the DSPMSG command to check the status of a dictionary being created:
  - a. On the command line, type DSPMSG.

When the dictionary is finished being created, the message Dictionary available in library ADSILEDInn appears when you type this command.

b. Press F3 (Exit).

The display from which you entered the DSPMSG command appears.

c. Press F12 (Cancel) until you have exited AppDict Services/400.

## What You Have Learned in Module 1

You have learned how to:

- Create a new dictionary, and populate the new dictionary with an existing library
- Check the status of the dictionary to determine when it has been fully created, and is ready to use

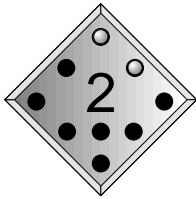
## Module 1 Questions

To review the information you learned in “Module 1: Create a New Dictionary to Contain the Program Modules” on page 17, try to answer the following questions. You will find the answers to these questions in “Module 1 Answers” on page 53.

1. The command you use to convert an existing library into a dictionary is the \_\_\_\_\_ *library-name* command.
2. A dictionary name can have up to \_\_\_\_\_ characters.
3. The first character of a dictionary name must be \_\_\_\_\_ .
4. The following characters are not allowed in dictionary names:  
\_\_\_\_\_ .
5. You know that a dictionary has been fully created, and is ready to use when the status shown for that dictionary in the Work with Dictionaries display is \_\_\_\_\_.
6. The command you can also use to confirm that a dictionary has been created is \_\_\_\_\_.

---

## Module 2: Familiarize Yourself with the Structure of the Sample Application



Often you are asked to make updates to programs that you have never worked on before. To feel more comfortable with the program before making changes to it, you should familiarize yourself with its structure.

AppDict Services/400 allows you to look at the structure of ILE programs through the Work with Bound Objects and Work with Binding Programs displays. The Work with Bound Objects display provides a list of the modules and service programs that are bound into the ILE program or service program that you are working with. Alternatively, the Work with Binding Programs display provides a list of the ILE programs and service programs that were created from binding the service program or module that you are working with.

The structure of the sample application is also illustrated in Figure 1 on page 15. You might want to refer to the flowchart while you are working through this module.

### Familiarize Yourself with the Bound Objects

To start, you can familiarize yourself with the modules and service programs that are bound into the ILE programs and service programs of the sample application.

To do this:

1. From an AS/400 command line, type the following:

```
STRADS ADSILEDI01
```

The Application Dictionary Services/400 menu appears.

2. From the Application Dictionary Services/400 menu, select 3 (Work with programs and modules), and press the Enter key.

The Subset Programs and Modules display appears.

3. Leave the defaults, \*ALL, in all of the prompts, and press the Enter key.

The Work with Programs and Modules display appears.

4. Type 19 (Work with bound objects) beside the service program MLGMTNS (the mailing list service program), and press the Enter key.

The Work with Bound Objects display appears.

```

Work with Bound Objects
Dictionary . . . . . : ADSILEDI01   Library . . . . . : ADSILEU01
Program . . . . . : MLGMTNS       Type . . . . . : *SRVPGM
Reference level . . . . . : +1       Attribute . . . . . :
Text . . . . . :

Type options, press Enter.
  2=Edit      4=Delete      5=Display      14=Compile
  19=Work with bound objects...

Opt Object      Library  Type      Attribute  Text
— MLGMTNC      ADSILEU01 *MODULE   CLLE       Mailing list maintenance
— MLGMTNR      ADSILEU01 *MODULE   RPGLE      Mailing list master mainten
— MLGNAMR      ADSILEU01 *MODULE   RPGLE      Mailing list name search
— QLEAWI       QSYS       *SRVPGM
— QLEAWI       QSYS       *SRVPGM   (Object not documented.)
— QLEAWI       QSYS       *SRVPGM   (Object not documented.)
— QRNXIE       QSYS       *SRVPGM   (Object not documented.)
— QRNXIO       QSYS       *SRVPGM   (Object not documented.)

Bottom
F3=Exit      F4=Prompt      F5=Refresh      F10=Command entry
F12=Cancel   F16=User options  23=More options  F24=More keys

```

Figure 4. Work with Bound Objects Display

Notice that some of the service programs are not documented in the dictionary. This is because these service programs are ILE run-time service programs that are automatically linked to the programs and service programs of an application by the ILE binder.

5. Press F12 (Cancel) to return to the Work with Programs and Modules display.
6. You can continue viewing the bound modules and service programs within the sample application by repeating steps 4 through 5.

## Familiarize Yourself with the Binding Programs

Next, you can familiarize yourself with the ILE programs and service programs that were created from binding MLGINQR, the mailing list inquiry module.

To do this:

1. From the Work with Programs and Modules display, type 16 (Work with binding programs) beside the MLGINQR module, and press the Enter key.

The Work with Binding Programs display appears.

```

                                Work with Binding Programs
Dictionary . . . . . : ADSILEDI01   Library . . . . . : ADSILEU01
Bound object . . . . . : MLGINQR     Type . . . . . : *MODULE
Reference level . . . . . : -1        Attribute . . . . . : RPGLE
Text . . . . . : Mailing list inquiry

Type options, press Enter.
 4=Delete      5=Display      9=Run          14=Compile
16=Work with binding programs...

Opt Program Library Type Attribute Text
— MAILLIST ADSILEU01 *PGM CLLE Mailing list menu program

F3=Exit      F4=Prompt      F5=Refresh      F10=Command entry
F12=Cancel   F16=User options F23=More options F24=More keys

                                Bottom

```

Figure 5. Work with Binding Programs Display

The display shows that the main program of the Mailing List sample application (MAILLIST) binds the mailing list inquiry module (MLGINQR).

2. Press F12 (Cancel) to return to the Work with Programs and Modules display.
3. You can continue viewing the binding programs and service programs within the sample application by repeating steps 1 through 2 for the other service programs and modules listed.
4. Press F12 (Cancel) to return to the Application Dictionary Services/400 menu.

## What You Have Learned in Module 2

You have learned how to familiarize yourself with the binding relationships of an ILE program by looking at the:

- Bound objects
- Binding programs

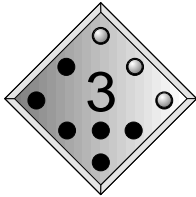
## Module 2 Questions

To review the information you learned in “Module 2: Familiarize Yourself with the Structure of the Sample Application” on page 21, try to answer the following questions. You will find the answers to these questions in “Module 2 Answers” on page 53.

1. The Work with Bound Objects display lists the \_\_\_\_\_ that are bound into the ILE program or service program that you are working with.
2. The Work with Binding Programs display lists the \_\_\_\_\_ that were created from binding the service program or module that you are working with.

3. To get to the Work with Bound Objects display, you type \_\_\_\_\_ from the Work with Programs and Modules display.
4. To get to the Work with Binding Programs display, you type \_\_\_\_\_ from the Work with Programs and Modules display.

## Module 3: Check the Current Field Length and Type



When you are approached with design changes, you often want to verify that the change you are being asked to make is valid. This module shows you how to check that the current length and type of the MLZIP field in the Mailing List program needs to be changed.

You check the current length and type of a field by:

- Locating the specific field you are interested in
- Locating the record (that contains the field) within the specific file
- Locating the field within the specified record, and displaying its length and type

### Locate the MLZIP Field

To locate the MLZIP field that you are interested in:

1. From the Application Dictionary Services/400 menu, select 1 (Work with fields).

The Subset Fields to Work With display appears, as shown in Figure 6.

```

                                Subset Fields to Work With
Dictionary . . . . . : ADSILEDI01
Type choices, press Enter.
Field . . . . . MLZIP*          *ALL, name, generic*
Record . . . . . *ALL           *ALL, name, generic*
File . . . . . *ALL             *ALL, name, generic*
Library . . . . . *ALL          *ALL, name, generic*
Attribute . . . . . *ALL        *ALL, attribute
                                F4 for list
Text:
Search words . . . . . _____
                                _____
Search condition . . . . . 1 1=Or, 2=And

F3=Exit      F4=Prompt      F5=Refresh      F12=Cancel
```

Figure 6. Subset Fields to Work With Display

2. In the Subset Fields to Work With display, type the following, and press the Enter key:
  - a. In the *Field* prompt, type MLZIP\*.
  - b. In the *Record*, *File*, *Library*, and *Attribute* prompts, specify \*ALL (the default), and press the Enter key.

The Work with Fields display appears.

```

                                Work with Fields
Dictionary . . . . . : ADSILEDI01
Position to . . . . . : _____ Starting characters of field

Type options, press Enter.
  2=Edit          5=Display          8=Work with impacted files
  9=Work with related files        12=Work with record formats ...

Opt  Field      Record   File      Library   Attribute
-   MLZIP      DETAIL   MLGNAMD   ADSILEU01 DSPF
-   MLZIP      DSPLY2   MLGINQD   ADSILEU01 DSPF
-   MLZIP      DSPLY2   MLGMTND   ADSILEU01 DSPF
-   MLZIP      MLGMSTR  MLGMSTL   ADSILEU01 LF
-   MLZIP      MLGMSTR  MLGMSTL2  ADSILEU01 LF
-   MLZIP      MLGMSTR  MLGMSTL3  ADSILEU01 LF
-   MLZIP      MLGMSTR  MLGMSTP   ADSILEU01 PF-DTA
                                           Bottom

Command
====>
F3=Exit          F4=Prompt   F5=Refresh   F6=Add fields
F9=Retrieve      F12=Cancel  F23=More options F24=More keys
This is a subsetted list.
```

Figure 7. Work with Fields Display, Showing the MLZIP Fields in the Mailing List Program

Do not press any more keys. Go on to the next section.

### Look at the Record Format Which Contains the Field Description

**Note:** This procedure follows from the previous section (“Locate the MLZIP Field”). The Work with Fields display should still appear on your screen.

To look at the record format of the MLZIP field:

1. Press the PageDown key to scroll to the next set of fields.
2. In the *Opt* prompt beside the MLZIP field displayed for the file MLGREFP in the Work with Fields display, type 12 (Work with record formats) as shown in Figure 8 on page 27, and press the Enter key.

```

Work with Fields
Dictionary . . . . . : ADSILEDI01
Position to . . . . . : _____ Starting characters of field

Type options, press Enter.
  2=Edit                5=Display                8=Work with impacted files
  9=Work with related files          12=Work with record formats ...

Opt Field      Record  File      Library  Attribute
12 MLZIP      MLGREFR  MLGREFP  ADSILEU01 PF-DTA
  _  MLZIP      SFLRCD   MLGNAMD  ADSILEU01 DSPF

Command Bottom
====>
F3=Exit      F4=Prompt      F5=Refresh      F6=Add fields
F9=Retrieve   F12=Cancel     F23=More options F24=More keys
This is a subsetted list.

```

Figure 8. Work with Fields Display, Showing the MLZIP Field in the MLGREFP File Being Selected

The Work with Record Formats display appears, listing the record formats used for this field, as shown in Figure 9.

```

Work with Record Formats
Dictionary . . . . . : ADSILEDI01
File . . . . . : MLGREFP Attribute . . . . . : PF-DTA
Library . . . . . : ADSILEU01
Text . . . . . : Mailing maintenance field reference file

Type options, press Enter.
  11=Work with fields in record format

Opt Record  Field Length Level ID      Text
  _  MLGREFR      8    79  5CD2461C2D490 Mailing list ref

Bottom
F3=Exit      F4=Prompt      F5=Refresh      F10=Command entry
F12=Cancel   F13=Repeat     F24=More Keys

```

Figure 9. Work with Record Formats Display, Showing the Record Formats in the MLGREFP File

You will be working with the MLGREFR record in the next section.  
*Do not press any more keys. Go on to the next section.*

## Locate the Field and Display Its Length and Type

**Note:** This procedure follows from the previous section (“Look at the Record Format Which Contains the Field Description”). The Work with Record Formats display should still appear on your screen.

To locate the field within the MLGREFR record, and display its length and type:

1. In the *Opt* prompt beside the MLGREFR record, type 11 (Work with fields in record format), and press the Enter key.

The Work with Fields in Record Format display appears.

Work with Fields in Record Format							
Dictionary . . . . .	:	ADSILEDI01					
File . . . . .	:	MLGREFP		Attribute . . . . .	:	PF-DTA	
Library . . . . .	:	ADSILEU01		Record . . . . .	:	MLGREFR	
Text . . . . .	:	Mailing list field reference file					
Type options, press Enter.							
5=Display		8=Work with impacted files					
20=Work with objects to recreate							
Opt	Field	Input offset	Output offset	Type	Key	Length	Decimal
—	MLACCT	1	1	P		5	0
—	MLADDR	35	35	A		20	
—	MLCITY	55	55	A		20	
—	MLNAME	5	5	A		20	
—	MLSRCH	25	25	A		10	
—	MLSTAT	75	75	A		2	
—	MLTYPE	4	4	A		1	
<b>Bottom</b>							
F3=Exit		F4=Prompt		F5=Refresh		F10=Command entry	
F11=Display text				F12=Cancel		F24=More keys	

Figure 10. Work with Fields in Record Format Display, Showing Fields in the MLGREFR Record

2. Using the PageDown key, scroll to the MLZIP field.

3. Look at the value for this field under the *Type* column.

It is type P. The P stands for packed decimal.

4. Look at the value for this field under the *Length* column.

It is 5 characters long.

This information verifies that the design request is valid. The field type would have to be changed to allow an alphanumeric postal code (zip) number, and the field length would have to be increased by 1 character to accommodate a 6-character postal code number.

5. Press F12 (Cancel) four times to return to the Application Dictionary Services/400 menu.

## What You Have Learned in Module 3

You have learned how to:

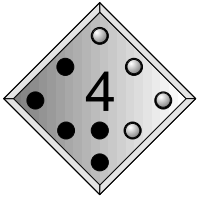
- Locate fields in a program
- Look at a record format for a field
- Locate a field within a specified record format, and display the type and length of the field

## Module 3 Questions

To review the information you learned in “Module 3: Check the Current Field Length and Type” on page 25, try to answer the following questions. You will find the answers to these questions in “Module 3 Answers” on page 53.

1. To locate a field, you type \_\_\_\_\_ on the Application Dictionary Services/400 menu.
2. The fields in a dictionary are listed in the \_\_\_\_\_ display.
3. The records in a specified file are listed in the \_\_\_\_\_ display, which appears when you type 12 (Work with record formats) from the \_\_\_\_\_ display.
4. The fields in a specified record format are listed in the \_\_\_\_\_ display, which is displayed by typing 12 (Work with fields in record format) from the \_\_\_\_\_ display.

## Module 4: Assess the Effect of the Design Change



This module shows you how to assess the effect that making a change to the MLZIP field will have on other objects by determining the:

- Fields referenced by the changing field
- Files that would be affected by the changing field
- Programs, service programs, and modules that would be affected by the field to be changed

### Determine Which Fields Would Be Affected

The AppDict Services/400 feature provides the capability to determine the effect that changing a field may have on other fields in the program by allowing you to view the **field reference hierarchy**. The field reference hierarchy is displayed as a list of fields, each field having a field reference value that shows the tree structure of the field references. The field reference hierarchy is shown in relation to the **base field**, which is the field with which you are working.

Using your knowledge of the field type and contents, along with the knowledge that AppDict Services/400 provides about the field reference hierarchy, you can decide whether or not to make the change, and plan how to update other programs and fields that will be affected.

Determine all of the fields referenced by the field to be changed as follows:

1. From the Application Dictionary Services/400 menu, type 1 (Work with fields), and press the Enter key.

The Subset Fields to Work With display appears as shown in Figure 11.

```

                                Subset Fields to Work With
Dictionary . . . . . : ADSILEDI01
Type choices, press Enter.
Field . . . . . MLZIP*      *ALL, name, generic*
Record . . . . . *ALL      *ALL, name, generic*
File . . . . . *ALL        *ALL, name, generic*
  Library . . . . . *ALL      *ALL, name, generic*
Attribute . . . . . *ALL      *ALL, attribute
                                F4 for list
Text:
  Search words . . . . . _____
                                _____
  Search condition . . . . . 1      1=Or, 2=And

F3=Exit      F4=Prompt      F5=Refresh      F12=Cancel
```

Figure 11. Subset Fields to Work With Display

2. Type in the data for the prompts as follows, and press the Enter key. (If you have not interrupted the self-study to do something else in AppDict Services/400, these will be the defaults.)

- a. In the *Field* prompt, type MLZIP\*.
- b. In the *Record, File, Library, and Attribute* prompts, type \*ALL.

The Work with Fields display appears, listing all of the MLZIP fields used in the Mailing List program, as shown below.

```

                                Work with Fields
Dictionary . . . . . : ADSILEDI01
Position to . . . . . _____ Starting characters of field

Type options, press Enter.
  2=Edit          5=Display          8=Work with impacted files
  9=Work with related files        12=Work with record formats ...

Opt  Field      Record   File      Library   Attribute
-   MLZIP      DETAIL   MLGNAMD   ADSILEU01 DSPF
-   MLZIP      DSPLY2   MLGINQD   ADSILEU01 DSPF
-   MLZIP      DSPLY2   MLGMTND   ADSILEU01 DSPF
-   MLZIP      MLGMSTR  MLGMSTL   ADSILEU01 LF
-   MLZIP      MLGMSTR  MLGMSTL2  ADSILEU01 LF
-   MLZIP      MLGMSTR  MLGMSTL3  ADSILEU01 LF
-   MLZIP      MLGMSTR  MLGMSTP   ADSILEU01 PF-DTA
                                          More...

Command
===>
F3=Exit          F4=Prompt      F5=Refresh     F6=Add fields
F9=Retrieve      F12=Cancel     F23=More options F24=More keys
This is a subsetted list.

```

Figure 12. Work with Fields Display, Showing the MLZIP Fields in the Mailing List Program

- 3. Press PageDown to scroll to the next set of fields displayed in the list.
- 4. To see which fields are directly or indirectly related to the MLZIP field we want to change:
  - a. Type 19 (Work with field references) beside the MLZIP field listed for the file MLGREFP, and press the Enter key. (This option does not display initially. You can display it by pressing F23 (More options) two times.)

The Work with Field References display appears, as shown in Figure 13 on page 32. This display lists all other fields in the dictionary that are related to the MLZIP field in the MLGREFP file (described at the top of the display), and provides information about the reference level of each of the related fields.

```

                                Work with Field References
Dictionary . . . . . : ADSILEDI01
Field . . . . . : MLZIP           Record . . . . . : MLGREFR
File . . . . . : MLGREFP         Library . . . . . : ADSILEU01

Type options, press Enter.
  2=Edit           5=Display           6=Design
 10=Work with programs referencing file  14=Compile

Opt  Level  Field      Record    File      Library    Attribute
---    ---      ---          ---         ---         ---         ---
---    0      MLZIP      MLGREFR     MLGREFP     ADSILEU01   PF-DTA
---    1      MLZIP      DETAIL      MLGNAMD     ADSILEU01   DSPF
---    1      MLZIP      DSPY2       MLGINQD     ADSILEU01   DSPF
---    1      MLZIP      DSPY2       MLGMTND     ADSILEU01   DSPF
---    1      MLZIP      MLGMSTR     MLGMSTP     ADSILEU01   PF-DTA
---    1      MLZIP      SFLRCD      MLGNAMD     ADSILEU01   DSPF
---    2      MLZIP      MLGMSTR     MLGMSTL     ADSILEU01   LF
---    2      MLZIP      MLGMSTR     MLGMSTL2    ADSILEU01   LF

                                                    Bottom
F3=Exit           F4=Prompt      F5=Refresh     F12=Cancel
F11=Display text  F24=More keys

```

Figure 13. Work with Field References Display. This display lists all of the MLZIP fields that are related to the MLZIP field in the MLGREFP file.

- b. Look at the value displayed in the *Level* column for the first MLZIP field. The 0 indicates that this field is a base field.
- c. Look at the value displayed in the *Level* column for the second MLZIP field. The 1 indicates that this field references the base field MLZIP (in the MLGREFR record).

The other values that can appear in the *Level* column are:

<b>Level</b>	<b>Description</b>
<b>N (where N &gt; 1)</b>	Indicates that the field makes a reference to a field that is ranked at level N-1.  For example, if a field has a <i>Level</i> value of 2 (value N), this would indicate that the field makes a direct reference to fields with a <i>Level</i> value of 1 (value of N-1).
<b>-N (where N &gt;= 1)</b>	Indicates that the field is referenced by a field that is ranked at level -(N-1). Fields with a level of -1 are fields from which the base field is, itself, derived. A field with a level of -1 may have other fields derived from it that are not shown on the Work with Field References display. Furthermore, these fields may themselves be derived from other fields.

Figure 14 illustrates the field reference level hierarchy for the MLZIP field:

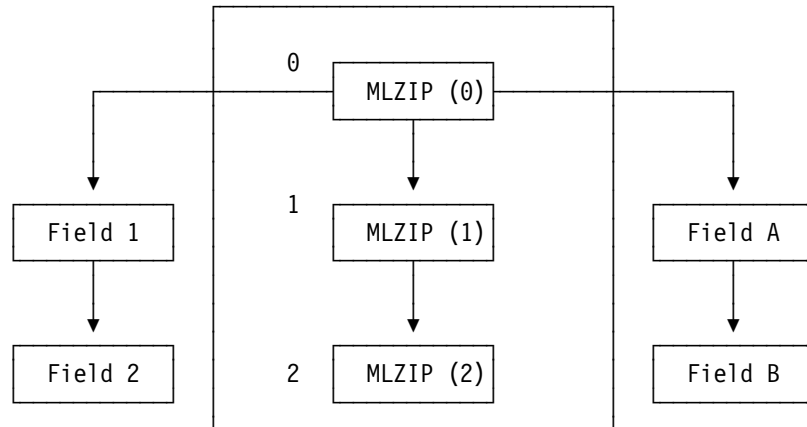


Figure 14. Relative Reference Levels of Fields

Each of the labeled boxes represents a hypothetical **field definition**—the descriptive information that AppDict Services/400 keeps for a field. The fields enclosed by the larger box represent the fields shown on the Work with Field References display, with the MLZIP field in the MLGREFR record selected as the base field. The connecting lines that run outside the larger box show how there may be other fields (not shown on the Work with Field References display) that reference the MLZIP field. Fields that have not been documented in AppDict Services/400, will not appear on the Work with Field References display.

- d. Press F12 (Cancel) three times to return to the Application Dictionary Services/400 menu.

## Determine Which Files Would Be Affected

To determine which files would be affected by changing the length of the MLZIP field:

1. From the Application Dictionary Services/400 menu, type 1 (Work with fields), and press the Enter key.

The Subset Fields to Work With display appears.

2. Type data in the display prompts as follows, and press the Enter key.

- a. In the *Field* prompt, type MLZIP\*.
- b. In the *Record, File, Library, and Attribute* fields, type \*ALL.

The Work with Fields display appears, listing all of the MLZIP fields used in the Mailing List program.

3. Press PageDown to scroll to the next set of fields displayed in the list.
4. Type 8 (Work with impacted files) beside the MLZIP field for the MLGREFP file,, and press the Enter key.

The Work with Impacted Files display appears, listing all of the files (documented in the dictionary) that would be affected by a change to the MLZIP field in the MLGREFP file.

```

                                Work with Impacted Files
Dictionary . . . . . : ADSILEDI01
Field . . . . . : MLZIP           Record . . . . . : MLGREFR
File . . . . . : MLGREFP         Library . . . . . : ADSILEU01

Type options, press Enter.
  2=Edit                4=Delete                9=Work with related files
 10=Work with programs referencing file    14=Compile

Opt  Level  File      Library  Attribute  Text
---  ---    ---      ---      ---        ---
 0    0    MLGREFP  ADSILEU01  PF-DTA    Mailing list field reference f
 1    1    MLGINQD  ADSILEU01  DSPF      Mailing list inquiry display
 1    1    MLGMSTP  ADSILEU01  PF-DTA    Mailing master file
 1    1    MLGMTND  ADSILEU01  DSPF      Mailing maintenance display fi
 1    1    MLGNAMD  ADSILEU01  DSPF      Mailing list name search displ
 2    2    MLGMSTL  ADSILEU01  LF        Mailing list label printing lo
 2    2    MLGMSTL2 ADSILEU01  LF        Mailing list by state, city, s
 2    2    MLGMSTL3 ADSILEU01  LF        General purpose LF for queryin
                                          More...
F3=Exit      F4=Prompt    F5=Refresh   F6=Work with objects to recreate
F12=Cancel   F21=Print list F24=More keys

```

Figure 15. Work with Impacted Files Display, Showing Files that Would Be Impacted

The *Level* field indicates the reference level for each file listed on the display. Table 4 lists all of the files that would be impacted by changing the MLZIP field.

Table 4. Files that Would Be Impacted By Changing the MLZIP Field

Reference Level	File Name	File Attribute	Description
0	MLGREFP	PF-DTA	Mailing list field reference file
1	MLGINQD	DSPF	Mailing list inquiry display
1	MLGMSTP	PF-DTA	Mailing master file
1	MLGMTND	DSPF	Mailing maintenance display file
1	MLGNAMD	DSPF	Mailing list name search display file
2	MLGMSTL	LF	Mailing list printing logical
2	MLGMSTL2	LF	Mailing list by state, city, search
2	MLGMSTL3	LF	General purpose LF for querying MLGMSTP
2	MLGNAML	LF	Mailing list logical by name, state, city

5. Look at the reference level for the MLGREFP file.

It is 0, which indicates that the MLZIP field is defined in this file.

The other possible values that the reference level can have are:

Level	Description
1	Indicates that a field in this file refers to the given field.
N	Indicates that a field in this file refers to another field in the file that has a <i>Level</i> value of N-1.

If a file has multiple reference levels in another file, the highest field reference level will be shown.

The reference level is used to determine the order in which the objects are compiled. For example, if the list of impacted files F0 through F3 displayed field reference levels of 0 through 3, the order in which those files would be recompiled is:

- 0            Recompile file F0
- 1            Recompile file F1
- 2            Recompile file F2
- 3            Recompile file F3

Because the MLZIP field has a field reference level of 0, in order to reflect the change in length of this field in the MLGREFP file, all of the files and programs that reference it need to be recompiled.

6. To return to the Application Dictionary Services/400 menu, press F12 (Cancel) three times.

## Determine Which Programs or ILE Objects Would Be Affected

After determining which files have been affected, you can determine which programs or ILE objects would need to be recompiled and recreated if any changes were made to the MLZIP field. To determine which programs or ILE objects would be affected if the MLZIP field length was changed:

1. From the Application Dictionary Services/400 menu, type 1 (Work with fields).  
The Subset Fields to Work With display appears.
2. Type data in the display prompts as follows, and press the Enter key.
  - a. In the *Field* prompt, type MLZIP\*.
  - b. In the *Record, File, Library, and Attribute* fields, type \*ALL.

The Work with Fields display appears, listing all of the MLZIP fields used in the Mailing List program.

3. Press PageDown to scroll to the next set of fields displayed in the list.
4. In the *Opt* prompt beside the MLZIP field displayed for the MLGREFP file, type 8 (Work with impacted files), and press the Enter key.

The Work with Impacted Files display appears.

5. From the Work with Impacted Files display, press F15 (Work with impacted programs).

The Work with Impacted Programs and Modules display appears, listing all the programs and ILE objects that refer to the files listed on the Work with Impacted Files display.

```

                                Work with Impacted Programs and Modules
Dictionary . . . . . : ADSILEDI01
Field . . . . . : MLZIP           Record . . . . . : MLGREFR
File . . . . . : MLGREFP         Library . . . . . : ADSILEU01

Type options, press Enter.
  2=Edit           4=Delete           14=Compile
 15=Scan RPG source 25=Find string

Opt Object      Library  Type      Attribute  Use  Text
— MLGINQR      ADSILEU01 *MODULE  RPGLE      03  Mailing list inquiry
— MLGLBLR      ADSILEU01 *MODULE  RPGLE      00  Mailing list label pri
— MLGMTNR      ADSILEU01 *MODULE  RPGLE      03  Mailing list master ma
— MLGNAMR      ADSILEU01 *MODULE  RPGLE      07  Mailing list name sear
— MLGRPTR      ADSILEU01 *MODULE  RPGLE      01  Mailing list one line
— MAILLIST     ADSILEU01 *PGM      CLLE       03  Mailing list menu prog
— MLGMTNS      ADSILEU01 *SRVPGM          07
— MLGRPTS      ADSILEU01 *SRVPGM  RPGLE      01

Bottom
F3=Exit           F4=Prompt       F5=Refresh
F6=Work with objects to recreate F12=Cancel      F24=More keys

```

Figure 16. Work with Impacted Programs Display, Showing the Programs and ILE Objects that Are Affected

6. Press F12 (Cancel) four times to return to the Application Dictionary Services/400 menu.

In some applications, changing a field *will* have an impact on other programs, service programs, or modules within the application. For such applications, you can view the calling hierarchy to determine any changes you may need to make to application's exit and entry points. Since you are working with a field reference file in this self-study, you do not need to review the entry and exit points of the programs and ILE objects.

## What You Have Learned in Module 4

You have learned how to determine the:

- Fields referenced by a field to be changed
- Files that would be impacted by a field to be changed
- Programs, service programs, or modules that would be impacted by a field to be changed

## Module 4 Questions

To review the information you learned in “Module 4: Assess the Effect of the Design Change” on page 30, try to answer the following questions. You will find the answers to these questions in “Module 4 Answers” on page 54.

1. The methods used in this module to assess the effect of changing the length of a field were to:
  - a. Determine the \_\_\_\_\_ referenced by the field to be changed.
  - b. Determine the \_\_\_\_\_ that would be affected
  - c. Determine the \_\_\_\_\_ that would be affected

2. To determine which fields are referenced by another field, you use option \_\_\_\_\_ in the Work with Fields display.
  
3. The levels displayed in the Work with Field References display indicate the reference level of a field (0, 1, -1, and N) as follows:
  - a. \_\_\_\_\_ indicates that the field is a base field.
  - b. \_\_\_\_\_ indicates that the field makes a direct reference to the base field.
  - c. \_\_\_\_\_ indicates that the base field is derived from this field.
  - d. \_\_\_\_\_ indicates that the field references a field ranked at level N-1.
  
4. To view the files that would be affected by a changed field, you use option \_\_\_\_\_ in the Work with Fields display.
  
5. The levels shown in the Work with Impacted Files display indicate the file reference levels (0, 1, and N) as follows:
  - a. \_\_\_\_\_ indicates that the field being assessed is defined in that file.
  - b. \_\_\_\_\_ indicates that a field in that file refers to the field being assessed.
  - c. \_\_\_\_\_ indicates that a field in that file refers to a field in a file with a *Level* value of N-1.
  
6. The file reference levels also indicate the order in which files would be recompiled if a field were changed. A file with a reference level of 0 is recompiled \_\_\_\_\_. All of the rest of the files are recompiled according to their reference level, which is ranked in \_\_\_\_\_ numerical order.
  
7. Programs that are affected by a changed field are displayed in the \_\_\_\_\_ display.
  
8. You can reach the Work with Impacted Programs and Modules display:
  - a. First, using option \_\_\_\_\_ from the Work with Fields display
  - b. Then, using function key \_\_\_\_\_ from the Work with Impacted Files display

---

## Module 5: Update the Current Field Reference File for Localization



To localize the Mailing List program, you might want to create another field reference file that contains a duplicate set of fields which you can modify to suit a particular country where your application will be used. However, for AppDict Services/400 to extract the impact-analysis information for the new file, all of the programs, service programs, modules, and files used in the sample application would have to be recreated or recompiled.

Since this self-study is only meant to show the basic functions of AppDict Services/400, this module will show you how to update the existing field reference file. Because AppDict Services/400 is integrated with other tools in Application Development ToolSet/400 (such as Source Entry Utility (SEU)), you can easily edit the data or source in your files.

This module will show you how to edit the MLGREFP file from within AppDict Services/400.

### Edit the MLGREFP File to Change the Length and Type of the MLZIP Field

To edit the MLGREFP file:

1. From the Application Dictionary Services/400 menu, type 1 (Work with fields), and press the Enter key.

The Subset Fields to Work With display appears.

2. Fill in the prompts as follows, and press the Enter key:

- a. In the *Field* prompt, type MLZIP\*.

- b. In the *Record, File, Library and Attribute* prompts, leave \*ALL, the default.

The Work with Fields display appears.

3. Press PageDown to scroll to the next set of fields displayed in the list.

4. In the *Opt* prompt beside the MLZIP field used in the MLGREFP file, type 2 (Edit), and press the Enter key.

The SEU Edit display appears.

5. At the command line, type B0T, and press the Enter key.

The file scrolls to the bottom, and the MLZIP field appears. The following line defines the size and type of the MLZIP field:

```
MLZIP          5  0
```

6. Replace the 5 with a 6, and replace the 0 with a space.

7. Comment out (put an \* in column 7 beside) the next record in the source file:

```
EDTCDE(X)
```

8. Press F3 (Exit).

The Exit display appears, and Y should be in the *Change/create member* prompt.

9. Press the Enter key again.  
The Create Physical File (CRTPF) display appears.
10. Press the Enter key.  
The Submit Job (SBMJOB) display appears.
11. Press the Enter key again.  
The Confirm Replace of Object display appears.
12. In the *Replace* prompt, replace the N with a Y, and press the Enter key.  
Another Confirm Replace of Object display appears letting you know that any data associated with the physical file will be deleted.
13. In the *Replace* prompt, replace the N with a Y, and press the Enter key.  
A message appears indicating that the member in the file was changed.
14. Press F12 (Cancel) two times to return to the Application Dictionary Services/400 menu.

## What You Have Learned in Module 5

You have learned how to localize an existing field reference file by modifying the length and type of a field using SEU from within AppDict Services/400.

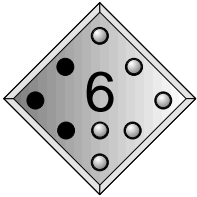
## Module 5 Questions

To review the information you learned in “Module 5: Update the Current Field Reference File for Localization” on page 38, try to answer the following questions. You will find the answers to these questions in “Module 5 Answers” on page 54.

1. To edit a field from the Work with Fields display, you type \_\_\_\_\_ beside the field you want to edit.
2. After you have updated a field in a file, you would need to recreate all of the affected \_\_\_\_\_.
3. AppDict Services/400 allows you to edit application objects by interfacing with \_\_\_\_\_.

---

## Module 6: Mark MLGREFP as a Field Reference File



This module describes how to mark the MLGREFP file you modified in “Module 5: Update the Current Field Reference File for Localization” on page 38 so AppDict Services/400 will recognize it as a field reference file.

AppDict Services/400 does not automatically recognize field reference files that are used in applications. You need to mark the MLGREFP field reference file so that when you use the functions that allow you to work with field reference files in AppDict Services/400, the file will be listed with all of the other field reference files.

### Mark the Field Reference Files

To mark the MLGREFP file as a field reference file:

1. From the Application Dictionary/Services menu, type 41 (Work with dictionaries), and press the Enter key.

The Work with Dictionaries display appears.

2. In the *Opt* prompt beside the ADSILEDI01 dictionary, type 10 (Mark field reference files), and press the Enter key.

The Mark Field Reference Files display appears listing the field reference files that need to be marked.

3. In the *Opt* prompt beside the MLGREFP file, type 1 (Select), and press the Enter key.

A message appears telling you the file has been marked as a field reference file, and the Work with Dictionaries display appears.

4. To exit, press F12 (Cancel).

The Application Dictionary Services/400 menu appears.

### What You Have Learned in Module 6

You have learned:

- That when you populate a dictionary with an existing library, you need to mark any field reference files contained in that library so that AppDict Services/400 will recognize them as field reference files
- How to mark a file as a field reference file

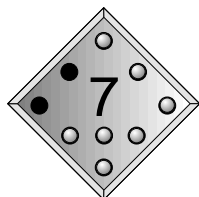
## Module 6 Questions

To review the information you learned in “Module 6: Mark MLGREFP as a Field Reference File” on page 40, try to answer the following questions. You will find the answers to these questions in “Module 6 Answers” on page 54.

1. Field reference files are marked from the \_\_\_\_\_ display.
2. You get to the Mark Field Reference Files display by typing option 10 (Mark field reference files) beside the \_\_\_\_\_ that documents the field reference file.
3. In the Mark Field Reference Files display you, type option \_\_\_\_\_ to select which files are to be marked as field reference files.

---

## Module 7: Search RPG Source for MLZIP Fields Not in Field Reference File



This module shows you how to search RPG source files for text strings. Because the Mailing List program might use MLZIP field definitions that are not defined in the field reference file, you should search the RPG source to find these other MLZIP field definitions, and update them with the new length and type. You only need to search ILE modules, since these are the only ILE objects that have source.

To search RPG source, AppDict Services/400 links with other tools in Application Development ToolSet/400. For example, AppDict Services/400 works with the Program Development Manager (PDM) to do the search, and Source Entry Utility (SEU) to allow you to edit the source files, if necessary.

Searches on AppDict Services/400 displays are **case-insensitive**. The search will match the text string that you type in the *Field* prompt, regardless of whether the letters are typed in upper or lower case. For example, if you type NEWYORK in the *Search words* field, the strings newyork, newYORK and NEWyork will also be found.

This module shows you:

- How to search RPG source files for text strings
- How AppDict Services/400 interacts with other Application Development ToolSet/400 tools during the search, such as PDM and SEU

### Search the RPG Source Files

To search the RPG source files for other MLZIP field definitions that need to be changed:

1. From the Application Dictionary Services/400 menu, type 3 (Work with programs and modules), and press the Enter key.

The Subset Programs and Modules to Work With display appears.

2. Type data in the prompts as follows, and press the Enter key.
  - a. In the *Library* prompt, type ADSILEU01.
  - b. In the other prompts, type \*ALL.

The Work with Programs and Modules display appears.

3. Type 25 (Find string) beside the *second* object (MLGINQR), and press the Enter key.

The Find String Using PDM display appears.

4. Type data in the prompts as follows (see Figure 17 on page 43), and press the Enter key:
  - a. In the *Find* prompt, type MLZIP.
  - b. In the *From column number* prompt, type 1.
  - c. In the *To column number* prompt, type \*RCDLEN to search up to the length of the record.
  - d. In the *Kind of match* prompt, type 2 to ignore the case.

- e. In the *Option* prompt, type 5 to display each source member that contains the search string.
- f. Leave the defaults in all of the rest of the prompts.

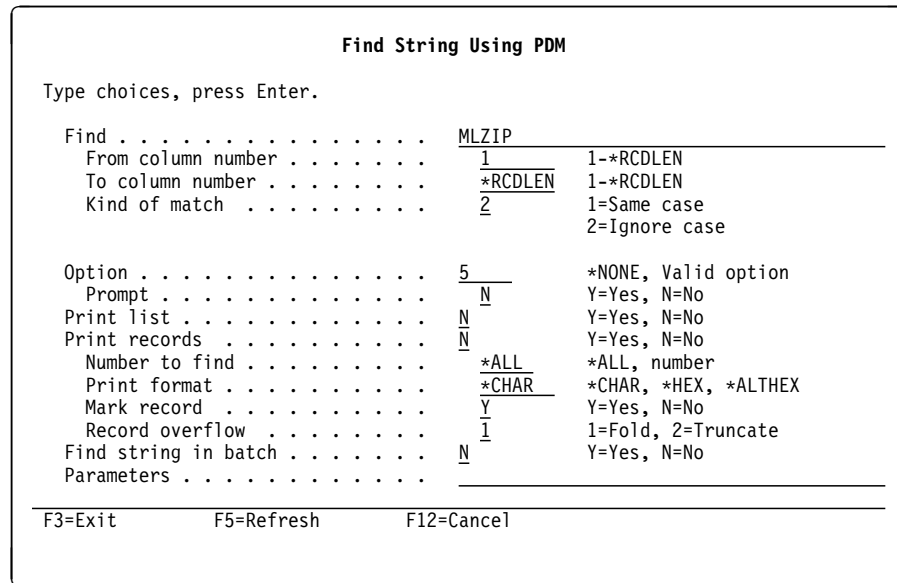


Figure 17. Find String Using PDM Display

A message appears indicating that a match was not found.

If a match had been found, the SEU Edit display would appear with the cursor on the first character of the MLZIP field found, and you would do the following:

- Edit the field to change the length to 6, and leave the type and the decimal position blank.
- Exit the SEU Edit display by pressing F3 (Exit).

The Work with Programs and Modules display would appear again.

5. Repeat steps 3 through 4 on page 42 for the rest of the source files listed in the Work with Programs and Modules display.

*You will not find any more MLZIP fields that need to be changed. Either their definitions are changed automatically by changing the field reference file, or their current definition can accommodate the increase in length and change in type.*

6. After you have finished searching the files, press F12 (Cancel) two times to return to the Application Dictionary Services/400 menu.

## What You Have Learned in Module 7

You have learned:

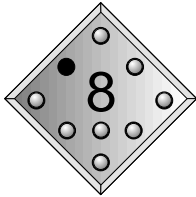
- How to search RPG source files for text strings
- How AppDict Services/400 interacts with other Application Development ToolSet/400 tools in the search, such as PDM and SEU

## Module 7 Questions

To review the information you learned in “Module 7: Search RPG Source for MLZIP Fields Not in Field Reference File” on page 42, try to answer the following questions. You will find the answers to these questions in “Module 7 Answers” on page 55.

1. AppDict Services/400 interacts with other Application Development ToolSet/400 tools such as \_\_\_\_\_ to allow you to edit your source code, and PDM to allow you to do things such as \_\_\_\_\_ .
2. You start the search function by typing option 25 (Find string) beside the first program listed in the \_\_\_\_\_ display.
3. You enter your search criteria in the \_\_\_\_\_ display.
4. When you specify \*RCDLEN in the *To column number* prompt, you indicate that you want to search \_\_\_\_\_ .
5. Specifying 5 in the *Option* prompt indicates that you want to display \_\_\_\_\_ .
6. When a text string is found that matches the search criteria, it is displayed in the \_\_\_\_\_ display.

## Module 8: Recreate All of the Changed Program Objects



This module describes how to recreate all of the objects that have been affected by changing the MLZIP field. The term **create** for ILE programs is synonymous with **compile** for Original Program Model (OPM) programs.

This module describes how to recreate all of the program parts affected by the changed field in one easy step.

### Recreate the objects Affected By the Modified MLGREFP File

To recreate the objects affected by the modified MLGREFP file:

1. From the Application Dictionary Services/400 menu, type 2 (Work with files).

The Subset Files to Work With display appears.

2. Type data in the prompts as follows, and press the Enter key.

- a. In the *File* prompt, type MLGREFP.
- b. In all of the other prompts, type \*ALL.

The Work with Files display appears.

3. In the *Opt* prompt beside the MLGREFP file, type 20 (Work with objects to recreate), and press the Enter key.

The Work with Objects to Recreate display appears.

```

                                Work with Objects to Recreate
Dictionary . . . . . : ADSILEDI01
Object . . . . . : MLGREFP                               Type . . . . : *FILE
Library . . . . . : ADSILEU01

Type options, press Enter.
2=Edit                               4=Drop
16=Change compile command

Opt  Object      Library    Type      Attribute  Text
---  ---
---  MAILLIST    ADSILEU01 *PGM      CLLE       Mailing list menu program
---  MLGINQD      ADSILEU01 *FILE     DSPF       Mailing list inquiry disp
---  MLGINQR      ADSILEU01 *MODULE   RPGLE      Mailing list inquiry
---  MLGLBLR      ADSILEU01 *MODULE   RPGLE      Mailing list label printi
---  MLGMSTL     ADSILEU01 *FILE     LF         Mailing list label printi
---  MLGMSTL2    ADSILEU01 *FILE     LF         Mailing list by state, ci
---  MLGMSTL3    ADSILEU01 *FILE     LF         General purpose LF for qu
---  MLGMSTP     ADSILEU01 *FILE     PF-DTA    Mailing master file

F3=Exit          F4=Prompt      F5=Refresh    F6=Submit recreate job
F11=Display more text  F12=Cancel     F24=More keys
                                More...

```

Figure 18. Work with Objects to Recreate Display, Showing the Objects that Will Be Recreated

4. To submit a batch job to recreate the objects in their libraries, press F6 (Submit recreate job).

The appropriate Create display appears for all of the objects one by one.

5. Press the Enter key for each Create display that appears.

A message appears indicating that a recreate job has been submitted.

Using the WRKSPLF (Work with Spooled Files) command, read the reports that appear in your output queue to verify that all of the affected objects displayed in the Work with Objects to Recreate display are recreated. Table 5 lists all of the objects that should have been recreated.

Table 5. List of Sample Application Objects that Should Have Been Recreated

Object	Type	Attribute
MAILLIST	*PGM	CLLE
MLGINQD	*FILE	DSPF
MLGINQR	*MODULE	RPGLE
MLGLBLR	*MODULE	RPGLE
MLGMSTL	*FILE	LF
MLGMSTL2	*FILE	LF
MLGMSTL3	*FILE	LF
MLGMSTP	*FILE	PF-DTA
MLGMTND	*FILE	DSPF
MLGMTNR	*MODULE	RPGLE
MLGMTNS	*SRVPGM	None
MLGNAMD	*FILE	DSPF
MLGNAML	*FILE	LF
MLGNAMR	*MODULE	RPGLE
MLGREFP	*FILE	PF-DTA
MLGRPTR	*MODULE	RPGLE
MLGRPTS	*SRVPGM	RPGLE

6. Press F12 (Cancel) three times to return to the Application Dictionary Services/400 display.

## What You Have Learned in Module 8

You have learned how to recreate all of the program parts affected by the changed file. In this case, the MLZIP field was updated, leaving the MLGREFP file changed.

## Module 8 Questions

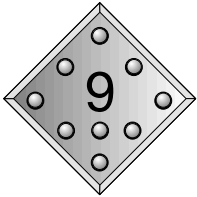
To review the information you learned in “Module 8: Recreate All of the Changed Program Objects” on page 45, try to answer the following questions. You will find the answers to these questions in “Module 8 Answers” on page 55.

1. When you make a change to a program object, and you want to prepare it so that it can be run again:
  - a. For ILE programs, you \_\_\_\_\_ it.
  - b. For OPM programs, you \_\_\_\_\_ it.

2. If a change to a field has an impact on other objects, you would recreate the other objects using option 20 (Work with objects to recreate) in the \_\_\_\_\_ display.
3. To submit the recreate job, you press \_\_\_\_\_ .

---

## Module 9: Run the Updated Sample Program



Now that you have completed the changes to the sample program, you should run it to ensure that it is still working correctly. This module shows you how to run a program from the Work with Programs and Modules display. You can also run programs from the Work with Calling Programs and Work with Called Programs displays.

### Run the Program

To run the sample program:

1. From the Application Dictionary Services/400 menu, type 3 (Work with programs and modules).

The Subset Programs and Modules to Work With display appears.

2. Fill in the prompts as follows, and press the Enter key:

- a. In the *Object* prompt, type MAILLIST.
- b. In the *Object Library*, *Type*, and *Attribute* prompts, type \*ALL.

The Work with Programs and Modules display appears.

3. In the *Opt* prompt beside the MAILLIST program, type option 9 (Run).

The program is run and the main menu of the Mailing List application appears.

4. If you were working in a real application development environment, you would test the application to ensure that it still works properly, and to assure yourself that changing the field has not caused any adverse effects.

### What You Have Learned in Module 9

You have learned how to run a program.

### Module 9 Questions

To review the information you learned in “Module 9: Run the Updated Sample Program,” try to answer the following questions. You will find the answers to these questions in “Module 9 Answers” on page 55.

1. You can run a program from the \_\_\_\_\_ , \_\_\_\_\_ , and \_\_\_\_\_ displays.
2. To run a program, you use option \_\_\_\_\_ .

---

## Chapter 4. Cleaning up the System

**Note:** Because the DELETE program attempts to delete the TEST user profile, you cannot call this program while you are using this profile.

After all of the students have completed the self-study and signed off, the exercise dictionaries, libraries, and user profiles should be deleted from your AS/400 system. *They should be deleted using the same user profile that created them prior to the self-study.* To clean up your AS/400 system:

1. Log on to the system.
2. On the command line, enter WRKAPPDCT.  
The Work with Dictionaries display appears.
3. Type 4 (Delete) in the *Opt* column next to the exercise dictionary ADSILEDI*nn* (for example, ADSILEDI01), and press the Enter key.  
The Confirm Delete of Dictionaries display appears.
4. To delete the exercise dictionary, press the Enter key again.  
The Work with Dictionary display appears again.
5. On the command line, enter CALL QDMTLAB/DELETE PARM(*nn*), where *nn* is the number of the user profile and library that you want to delete. For example, if you want to delete user profile ADS01, you would type CALL QDMTLAB/DELETE PARM(01).  
The library, and user profile that you specified is now deleted from the AS/400 system.
6. Complete steps 3 through 5 to delete the rest of the self-study dictionaries, libraries, and user profiles on your system.







---

## Appendix A. Answers to the Module Questions

---

### Module 1 Answers

Here are the answers to the questions in “Module 1 Questions” on page 20.

1. STRADS
2. 10
3. alphabetic
4. blank  
quotation mark
5. Not in use
6. DSPMSG

---

### Module 2 Answers

Here are the answers to the questions in “Module 2 Questions” on page 23.

1. modules and service programs
2. ILE programs and service programs
3. 19 (Work with bound objects)
4. 16 (Work with binding programs)

---

### Module 3 Answers

Here are the answers to the questions in “Module 3 Questions” on page 29.

1. 1 (Work with fields)
2. Work with Fields
3. Work with Record Formats  
Work with Fields
4. Work with Fields in Record Formats  
Work with Record Formats

---

## Module 4 Answers

Here are the answers to the questions in “Module 4 Questions” on page 36.

1. fields  
files  
programs, service programs, and modules
2. 19 (Work with field references)
3. 0  
1  
-1  
N
4. 8 (Work with impacted files)
5. 0  
1  
N
6. first  
ascending
7. Work with Impacted Programs and Modules
8. 8 (Work with impacted files)  
F15 (Work with impacted programs)

---

## Module 5 Answers

Here are the answers to the questions in “Module 5 Questions” on page 39.

1. 2 (Edit)
2. objects
3. SEU

---

## Module 6 Answers

Here are the answers to the questions in “Module 6 Questions” on page 41.

1. Work with Dictionaries
2. dictionary
3. 1 (Select)

---

## Module 7 Answers

Here are the answers to the questions in “Module 7 Questions” on page 44.

1. SEU  
search your source code for a text string
2. Work with Programs and Modules
3. Find String Using PDM
4. to the end of each record
5. each source member that contains the search string
6. SEU Edit

---

## Module 8 Answers

Here are the answers to the questions in “Module 8 Questions” on page 46.

1. recreate  
recompile
2. Work with Files
3. F6 (Submit recreate job)

---

## Module 9 Answers

Here are the answers to the questions in “Module 9 Questions” on page 48.

1. Work with Calling Programs  
Work with Called Programs  
Work with Programs and Modules
2. 9 (Run)



---

## Bibliography

This bibliography lists a variety of books that may be of use or interest to you as you work with the Application Dictionary Services/400 feature.

The Application Dictionary Services/400 library consists of the following publications:

- *ADTS/400: Application Dictionary Services/400 Self-Study Guide*, SC09-2086
- *ADTS/400: Application Dictionary Services/400 User's Guide*, SC09-2087

The Application Development ToolSet/400 library consists of the following publications:

- *ADTS/400: Advanced Printer Function*, SC09-1766
- *ADTS/400: Character Generator Utility*, SC09-1769
- *ADTS/400: Data File Utility*, SC09-1773
- *ADTS/400: File Compare and Merge Utility*, SC09-1772
- *ADTS/400: Interactive Source Debugger*, SC09-1897
- *ADTS/400: Programming Development Manager*, SC09-1771
- *ADTS/400: Report Layout Utility*, SC09-1767
- *ADTS/400: Screen Design Aid*, SC09-1768
- *ADTS/400: Source Entry Utility*, SC09-1774
- *Introducing ADTS/400 and AS/400 Server Access Programs*, GC09-2088
- *LPS: Application Development ToolSet for OS/400*, GC09-2089

The Application Development Manager/400 library contains the following publications:

- *ADTS/400: Application Development Manager/400 API Reference*, SC09-2180
- *ADTS/400: Application Development Manager/400 Introduction and Planning Guide*, GC09-1807
- *ADTS/400: Application Development Manager/400 User's Guide*, SC09-2133
- *ADTS/400: Application Development Manager/400 Self-Study Guide*, SC09-2138

The following publications in the AS/400 library may be of interest to you in relation to this feature:

- *Publications Reference*, SC41-4003
- *Backup and Recovery – Basic*, SC41-4304
- *CL Programming*, SC41-4721
- *CL Reference*, SC41-4722
- *Data Management*, SC41-4210
- *DB2 for OS/400 Database Programming*, SC41-4701
- *DB2 for OS/400 SQL Programming*, SC41-4611
- *DB2 for OS/400 SQL Reference*, SC41-4612
- *Experience RPG IV*, SC09-1938
- *ILE Application Development Example*, SC41-3602
- *ILE Concepts*, SC41-4606
- *Programming Reference Summary*, SX41-4720
- *Software Installation*, SC41-4120
- *System Operation*, SC41-4203
- *Work Management*, SC41-4306



---

# Index

## A

- adding
  - library list entry 16
- ADDLIBLE command 16
- ADSnn user ID, creating 16

## B

- batch
  - creating a dictionary 19
  - creating objects in dictionary 46
  - recreating objects 45

## C

- changing
  - fields 42
- changing fields
  - impacted files 33
  - impacted programs 35
- commands
  - ADDLIBLE 16
  - CALL with DELETE parameter 49
  - Create Program (CRTPGM) command
    - ENTMOD parameter 6
    - EXPORT parameter 6
  - Create Service Program (CRTSRVPGM) command
    - ENTMOD parameter 6
    - EXPORT parameter 6
  - Display Message (DSPMSG) 19
  - DSPMSG (Display Message) 19
  - RSTLIB 16
- compile order of objects 35
- convention used in this book viii
- Create Program (CRTPGM) command
  - ENTMOD parameter 6
  - EXPORT parameter 6
- Create Service Program (CRTSRVPGM) command
  - ENTMOD parameter 6
  - EXPORT parameter 6
- creating
  - ADSnn user ID 16
- creating a dictionary
  - specifying a name 17
  - verifying build 19

## D

- data items, sharing in ILE 6
- deleting
  - exercise libraries 49
  - exercise user ID 49

- determining
  - impacted files 33
  - impacted programs 35
- dictionary
  - creating 17
  - naming rules 17
- Display Message (DSPMSG) command 19
- displaying messages 19
- DSPMSG (Display Message) command 19

## E

- editing source with SEU 42
- ENTMOD parameter 6
- EXPORT parameter 6

## F

- field reference level description 32
- fields, changing 42
- file reference level description 34

## I

- ILE
  - See Integrated Language Environment (ILE)
- impacted files
  - determining 33
  - recreating 45
- impacted programs
  - determining 35
  - recreating 45
- Integrated Language Environment (ILE)
  - benefits 3
  - calling other objects
    - description 7
    - dynamic calls 7
    - static calls 7
  - data items
    - sharing between objects 6
  - definition 3
  - modules
    - described 5
    - procedures, contrasted with 5
  - object types for 3
  - OPM compared 3
  - procedures, described 4
  - programs
    - creating 6
    - programs, described 4
    - service programs
      - creating 6
      - described 5

## M

messages, displaying 19  
modules  
    ILE 5

## N

naming rules for dictionaries 17

## O

object types, ILE 3  
objects, compile order 35

## P

parameters  
    ENTMOD 6  
    EXPORT 6  
procedures  
    ILE 4, 5  
programs  
    creating  
        ILE 6  
programs, ILE 4  
publications, related 57

## R

recreating  
    impacted files 45  
    impacted programs 45  
recreating objects after changing a field 45  
relative reference levels of fields, example 33  
removing  
    exercise dictionaries 49  
restoring  
    library 16  
    self-study library 16  
RSTLIB command 16

## S

self-study library  
    deleting 49  
    restoring 16  
service programs  
    creating  
        ILE 6  
    described 5  
    public interface to 5  
SEU  
    See source entry utility (SEU)  
source entry utility (SEU)  
    editing DDS source members 42

submit to batch  
    creating a dictionary 19  
    creating objects in dictionary 46  
    recreating objects 45

## U

user ID  
    creating ADSnn 16  
    deleting exercise user ID 49  
using  
    SEU 42