Note!

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

First Edition (September 1994)

This edition applies to Version 3, Release 1, Modification Level 0, of Application Development Manager/400 (Feature 1613), a feature of IBM Application Development ToolSet/400 (Program 5763-PW1) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address 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 nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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, 208 Harbor Drive, Stamford, Connecticut, USA 06904-2501

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

At the time of printing, this book contains references to products that are announced, but may not yet be available. These products will be available later in this release.

Trademarks

The following terms, denoted by an asterisk (*), used in this publication, are trademarks or service marks of International Business Machines Corporation in the United States or other countries:

Application System/400	AS/400
C/400	COBOL/400
DB2/400	IBM
IBMLink	ILE
Integrated Language Environment	Operating System/400
OS/400	PROFS
RPG/400	SQL/400
SystemView	400

About This Book

The information in this book is intended for anyone wanting to understand the basic concepts and the planning needed to make effective use of the IBM* Application Development Manager/400, a feature of the Application Development ToolSet/400 (ADTS/400) licensed program product (5763-PW1). This feature provides the means for efficiently and effectively managing the change that occurs during the application development cycle, in an Application System/400* (AS/400*) environment.

This book begins with an overview of what this feature does, who its users are, and the advantages it offers to both large and small application development teams. It also provides a description of the relationship between Application Development Manager/400 and other licensed programs in the AS/400 system. The remaining chapters then discuss the planning needed to take advantage of the functions this feature brings to a development environment.

Who Should Use This Book

This book is intended for:

- Enterprise managers who are interested in evaluating this feature—with a view to adding it to the tool kit their application development team uses, or to replacing an outmoded tool with the functions the Application Development Manager/400 feature provides.
- Members of an application development team—project leaders, testers, planners, and application developers—who are looking for information on how to plan the work they must do to set up a development environment and begin to use this feature.

The Application Development Manager/400 feature has been specifically designed to facilitate the work that application developers do while developing applications in an AS/400 environment. This book therefore assumes that readers are familiar with the AS/400 environment and its various interfaces. This book does not include descriptions of how to perform the specific application development tasks. Such tasks are described in detail in the *ADTS/400: Application Development Manager/400 User's Guide*.

The Sample Project

This book uses a sample project based on a payroll application to illustrate the planning that needs to be done and ways to create a functioning application development environment. However, how the functions of the Application Development Manager/400 feature are implemented depends on an enterprise's particular needs and wants, its processes, and the nature of the application it plans to develop. The sample project, as it is used in the Application Development Manager/400 library, illustrates just one approach a development team might take.

Definitions

This guide does not contain a glossary. New terms are defined where they first appear, and highlighted in **boldface** type. For easy reference, each definition is also indexed under the term itself and under the index entry entitled *definitions*.

For More Information

Two IBM publications describe the Application Development Manager/400 feature, this book, and *ADTS/400: Application Development Manager/400 User's Guide*.

In addition, all developers have access to online help for the Application Development Manager/400 CL commands. If they are using any one of the tools that is part of the Application Development ToolSet/400 licensed program, they also have access to the help that has been added to some of the programming development manager displays.

The online information consists of:

- Help for CL commands—prompt displays for each command, *contextual* help, which explains the field on which the cursor is positioned, and *extended* help, which explains the overall purpose of the command.
- Help for the displays—contextual help, which explains the field on which the cursor is positioned, and extended help, which explains the overall purpose of the display.

See the "Bibliography" on page 47 for a list of publications that are most relevant to this feature. For a complete list of all the IBM books in the AS/400 library, see the *Publications Reference*, SC41-3003.

Chapter 1. Introduction and Overview

Most programs today consist of many pieces of code, or components. A variety of products, of which the Application Development Manager/400 feature is but one, have been created to support the traditional project management activities of application development and the efficient management of one or more versions of the components that make up an application. This management applies to code in its source and object forms as it is developed, tested, and changed over the application development cycle and throughout the life of the application.

The Software Development Environment Today

Today, change is intrinsic to most software engineering endeavors. Applications consist of many components, some of which are based on previous versions, and all of which may be undergoing constant revision. The problems of application development are further complicated by the interdependencies that can exist between components. Changing just one component may result in changes to several other components that depend on it.

To build and maintain high-quality applications efficiently, application development organizations must have a consistent and systematic approach to managing the changes they are making to their applications. They are looking for standard methods and procedures and, where possible, automation to improve productivity and reduce backlogs.

Development organizations need to be able to organize and manage all the components of an application as a unit. They also need to be able to control the baseline or master version of an application. They often want to control multiple versions of components or of entire applications. And they want to be able to make quick fixes to the code they are developing. They need a mechanism that allows for shared access to components, and they want the means to plan and manage their entire development process. They must reduce the time they spend maintaining applications so that they have more time to develop new ones.

Tools that go by many different names have evolved to meet some or all of these requirements.

- *Configuration managers* that identify, manage, and control all source and object code in some consistent and efficient way through some or all of the software life cycle
- *Change managers* that address system maintenance problems
- *Version control* tools that store multiple versions of an object (along with information about each version)
- *Library systems* or *services* that provide version and configuration management and certain compilation facilities

What the Application Development Manager/400 Feature Does

The Application Development Manager/400 feature answers many of the needs of today's application developers. It provides a team of application developers, working in an AS/400 environment, with a mechanism for efficiently and effectively managing their development environment and its application objects throughout the life of the application. An application development team using this feature can:

- Define a flexible environment where production, follow-on, and maintenance versions of an application can be managed simultaneously.
- Organize several developers working on the same application.
- Build (or compile) an application quickly and easily, based on the components that have changed and the relationships between these components and all other components, and compile the components in the correct order, with a guarantee of no more level checks.
- Create and maintain several **versions** of an application. (A version can be a separate program or release that is either new or based on an existing application and that contains significant new code or function.)
- Choose between two user interfaces—the menu-driven programming development manager utility that is part of the Application Development ToolSet/400 licensed program, or the control language (CL) command interface.

Its Benefits

The Application Development Manager/400 feature brings many advantages to an application development environment; advantages that help a team of application developers to improve the quality and integrity of their applications, and that let them produce these applications more quickly.

- **A standard development process**

  A development team can define the application development environment that suits its organization and methods. This feature supports processes already in place; it does not impose another process.

- **Increased productivity**

  This feature organizes both the developers writing the code and the code to be written. As developers write their code and compile and test it, they work efficiently and productively in a well-organized development environment where changes to their code are managed.

- **Flexibility and versatility**

  The structure defined at the beginning of a project does not restrict the development team. This structure can be changed and refined at any time, as the demands of the project change. Developers can be added to or removed from the project, and code can be shared and reused.

- **Protection of investment in skills and existing applications**

  Entire applications or single components can be imported into this development environment, allowing an enterprise to capitalize on the skills of the development team and on the work already in place.

- **A tried and proven user interface**

  Developers can work in an environment with which they are already familiar. Whether they choose to work with the CL commands or through the programming development manager (PDM) interface, they do not have to invest time in learning a new environment. If option 2 of the Application Development ToolSet/400 product is installed, developers can also use the source entry utility (SEU), the screen design aid (SDA), the report layout utility (RLU), and the data file utility (DFU).

- **Support for several versions of an application**

  Developers can create and maintain multiple versions of an application in both the Application Development Manager/400 development environment and in a production environment. They can easily identify which versions of source and objects belong to a particular version of the application.

- **An automated build process**

  Developers can rely on the powerful build process to build, or compile, the source code for an application more quickly. They no longer have to analyze the relationships between pieces of code; the build process does this for them automatically.

- **Data security and integrity**

  The structure that the Application Development Manager/400 environment provides ensures the integrity and security of production, test, and development versions of the code. Developers are able to work with different versions of the code. And they work in an environment where they are assured that they cannot overwrite one another's changes.

- **An audit trail**

  A project log records what has changed in the application, the commands used to change the hierarchy or components it contains, who issued the commands, and when the activity took place. A project log report can be tailored to show information about one developer or several and for a specific time period.

- **Support for applications written in several AS/400 programming languages**

  The AS/400 programming languages supported today are RPG/400*, COBOL/400*, C/400*, DB2/400*, control language (CL), data description specifications (DDS), command source, Integrated Language Environment* (ILE*) C/400, ILE COBOL/400, ILE RPG/400, ILE SQL/400 C, ILE SQL/400 RPG, ILE SQL/400 CBL, ILE CL, user interface manager (UIM), and with user-defined types, all unsupported compilers that generate *PGM objects. Parts of type CSRC and PGM of language C or SQLC cannot be built.

- **User-defined types for AS/400 objects and new programming languages**

  User-defined types are part types users define that are recognized in the Application Development Manager/400 environment. Although this feature supports many AS/400 object types, a development team may be working with AS/400 objects that are not supported directly. User-defined types allow for the management of multiple versions of these objects. New part types can also be defined for source file members.

- **Notification about the status of any component of the application**

  Developers receive messages that tell them that a part they are requesting is already being changed, that it exists in another branch of the project hierarchy, and that a change made in one version may also have to be made in the other version. This is useful when fixes to a production version of a part have to be propagated to the follow-on version of the part. The person doing this work needs to know where the part is in the hierarchy and who has checked it out to a development group.

- **Ability to package applications**

  The Application Development Manager/400 feature provides a mechanism that automates the packaging of applications through the use of functions of the SystemView* System Manager/400 licensed program.

How a Development Team Uses This Feature

The two main types of users for this feature are: **project administrators** who define and administer the environment in which application developers work; and **application developers** who develop code while working within this development environment.

The Project Administrator

The person who creates a project is automatically authorized to work on the project as a project administrator. This person defines the phases, such as development and test, through which components of an application go before they are actually placed in a production environment. Every project must have a project administrator. The person with project administrator authorization to a project may, however, also work on that project as a developer or tester or as the project leader. In many cases, it may be the project leader who is the person with project administrator authorization. (It should be noted here that project administration in the context of the Application Development Manager/400 feature is not usually a full-time job, unlike system administration in the AS/400 environment.)

A person with access to the Application Development Manager/400 environment as a project administrator does the following tasks:

- Defines the development environment for an application and the development phases through which each component goes
- Enrolls application developers to the project, and other project administrators if required
- Maintains the development environment by:
  - Redefining a project hierarchy as work on a new version begins, and as new groups are needed for development and maintenance
  - Enrolling or removing developers as people join or leave the project
- Reclaims information about a project after the system has been restored.

The Application Developer

The application developer can be any member of the development team who has been given access to the application by the project administrator. Application developers usually only have update access to specific work areas. They do the following tasks:

- Create or change source code and other components of the application in a private work area (components to be modified can be copied from applications already under the control of this feature or from AS/400 libraries)
- Compile components, or applications, using the build process that automates much of this work
- Test components, or applications, either within the control of this feature or by moving the code to a test or production environment outside Application Development Manager/400 control.

Other Members of the Development Team

Some tasks are not specifically either administrator or developer tasks. Who does such work depends on the processes being followed, the demands of the application being developed, how a project is defined, and the access various members of the development team have to it. Any member of a development team who has the appropriate authorization to the Application Development Manager/400 environment can do the following tasks:

- Import an existing application from an AS/400 library to be under Application Development Manager/400 control. The project leader may be the person who does this. Developers are more likely to import one or several components.
- Test components, or the entire application as it nears completion.
- Test and build the *entire* application once it has been imported. Again, the project leader may be the one who does this task. Members of the test team may be the people who do this.
- Export a version of the application (source, objects, or both) from the development environment to a production or a test environment. This is another task the project leader may do.

Chapter 2. Relationship to the AS/400 Environment

The Application Development Manager/400 feature provides a development environment where application developers can work efficiently and productively, and capitalize on their expertise using existing development processes. It is a separately orderable feature of the Application Development ToolSet/400 licensed program product that is part of the application environment within the AS/400 system. Figure 1 illustrates the relationships between the Application Development Manager/400 feature and the AS/400 system and the Operating System/400* (OS/400*), SystemView* System Manager/400, Application Development ToolSet/400, CoOperative Development Environment/400 (CODE/400) licensed program products, and Application Dictionary Services/400, a feature of the Application Development ToolSet/400 product.

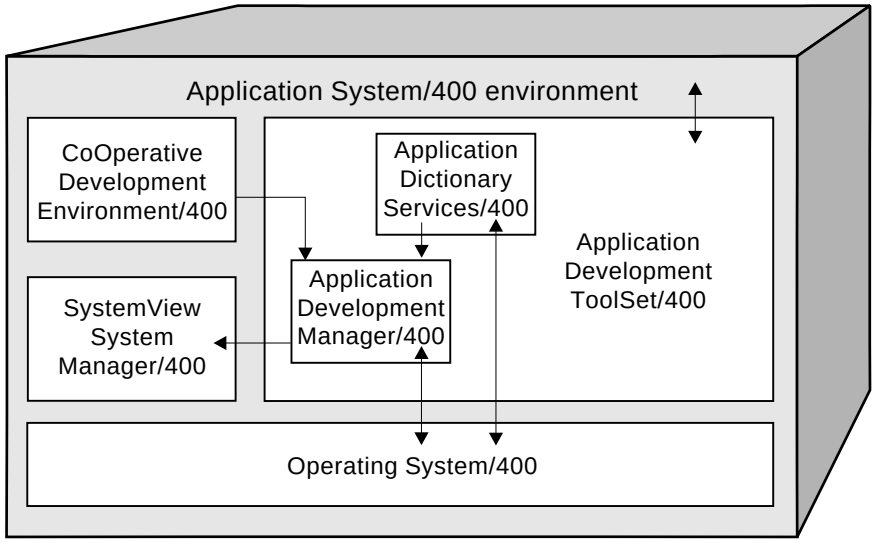

Figure 1. The Application Development Manager/400 feature in the AS/400 environment

- The AS/400 system, which consists of a family of mid-range computers based on a single software architecture, controls the operation of programs and provides services such as controlling resources, scheduling jobs, controlling input and output, and managing data.
- The OS/400 licensed program complements and extends the capabilities of the AS/400 system by providing fully integrated support for interactive applications. OS/400 functions are accessed through menus or control language (CL) commands.
- The Application Development ToolSet/400 licensed program product provides additional functions that integrate all program development services into a single environment.

  The programming development manager (PDM) utility, which is one component of this licensed program, provides an easy-to-use menu interface and access to a variety of other tools, such as, SEU, RLU, and DFU. This flexibility lets programmers move between the phases of program development, such as editing, compiling, and creating reports, all the while working within the Application Development Manager/400 environment.

- The System Manager/400 licensed program supports the Application Development Manager/400 feature, which provides the means whereby applications can be packaged using System Manager/400 product.
- From CODE/400 and Application Dictionary Services/400, developers also have access to the functions of the Application Development Manager/400 feature.

The Application Development Manager/400 feature interacts with all these products. It provides the mechanism that helps a team of developers to manage both their development environment and the applications they are developing. Developers access OS/400 functions through Application Development Manager/400–specific CL commands or through the user interface delivered by the programming development manager. And, when an application is complete, the Application Development Manager/400 feature works with System Manager/400 to package the application for distribution.

For more information about this environment, refer to the publications listed in the “Bibliography” on page 47.

Chapter 3. Planning the Application Development Environment

This chapter introduces the planning that is needed—as it relates to the principal tasks various members of the development team perform—to make the most effective use of the power and function of the Application Development Manager/400 feature.

Figure 2 illustrates the five basic steps to be completed before a team of application developers can begin the work of developing an application. If a new application for which no code exists is to be created, it is ready when the first three steps are complete. A new application requires that a project and the groups in it be created, and that the people who will be developing the code be enrolled in that project.

All five steps must be complete if the intent is to enhance an existing application. The fourth step brings the existing application under the control of the Application Development Manager/400 feature. Once there, all the components that have been imported must be compiled so that Application Development Manager/400 has a record of all the interrelationships between components of the application and to ensure that all components work as expected.

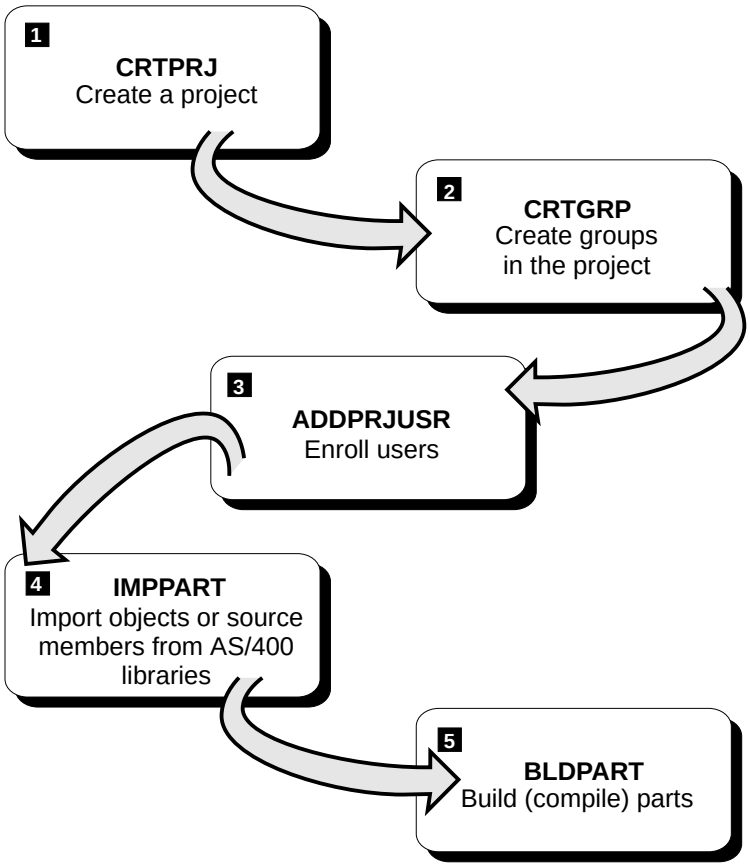

Figure 2. The steps to complete to begin working with this feature

To create this development environment and enroll the various members of the team to it, you must have a clear idea of how the development team will work on the application. Understanding how your application development team will work in this environment will enable you to optimize your current investment in the skills of your developers and in existing applications and to take advantage of the many benefits described in Chapter 1. You will need answers to many questions, of which the following are but a few.

- Will your application be stored in a single project, or will it be made up of several smaller projects?
- Will the developers be working on an existing application, or will they be starting something new?
- How many developers will work on the application?
- What are the phases in your current development process?
- Who will develop the code, and who will test it?
- How many levels of testing will you need?
- Who will administer the project?
- What will the role of the team leader be?
- Will pieces of code in the application be shared with other projects?
- How will the application be maintained?
- How many versions of an application will be supported?
- What level of security will be required?
- Will the application support more than one language?

Bear in mind that these considerations may vary greatly from project to project, and from development team to development team. The considerations presented in this book reflect the basic decisions that are common to most development projects.

What to Do Next

As you read the following chapters and consider the questions presented there, use the planning worksheets included in Chapter 9, “What to Do Next” to record the needs of your development team. When you move to the *ADTS/400: Application Development Manager/400 User’s Guide*, having this information at hand will facilitate the work of creating the Application Development Manager/400 development environment.

Chapter 4, “Planning a Project” describes an Application Development Manager/400 project and the questions you should consider before you start to use the Create Project (CRTPRJ) command.

Chapter 5, “Planning a Project's Hierarchy” discusses the groups that make a project hierarchy and the questions you should consider before you use the Create Group (CRTGRP) command.

Chapter 6, “Planning Enrollment” discusses the task of enrolling all the members of the development team to a project and the questions you should consider before you use the Add Project User (ADDPRJUSR) command.

Chapter 7, “Planning to Import an Application” discusses the task of importing an application and the questions you should consider before you use the Import Part (IMPPART) command.

Chapter 8, “Planning to Build an Application” describes the strength and versatility of the build process and the questions you should consider before you use the Build Part (BLDPART) command.

Chapter 9, “What to Do Next” directs you to the *ADTS/400: Application Development Manager/400 User’s Guide* and points out where you will find the information that relates to the planning you do as you answer the questions this book presents. This chapter also collects the questions discussed in Chapter 4 through Chapter 8 into a series of worksheets. You are encouraged to make notes there that will help you to create the development environment that suits your application and the team of developers who will develop it.

Chapter 4. Planning a Project

In the Application Development Manager/400 environment, a **project** is a complete application consisting of a collection of one or more groups. A **group** in a project is a collection of parts at the same stage in the development process. (A group can also be described as an AS/400 library under the control of this feature.) A **part** is an object, such as an AS/400 physical file or a program, a source member containing RPG/400 code for an RPG program, or some other item, such as a field in a record or a record in a file that you have defined for your application.

It is the concept of a project that lets you refer collectively to all the related components that make up one application, regardless of which development phase the components are in. A project is further defined as having a **hierarchy**, where its collection of groups is organized into levels, with each level representing a phase in the development process. One such collection of groups organized into the three levels common to most application development processes is illustrated in Figure 3. This concept of a project hierarchy provides the versatility, flexibility, and data security that application developers need to help them manage their day-to-day work.

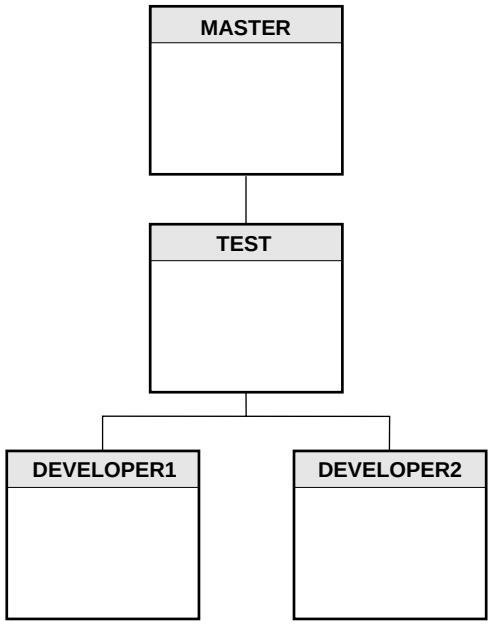

Figure 3. A sample project hierarchy

Questions to Consider

How you define a project and its hierarchy depends on the needs of your application and the processes your development team intends to use. Before you begin the actual work of creating a project, you should have answers to the following questions.

- **What will the project contain?**

  In most cases, one Application Development Manager/400 project contains all the components of a single application. However, there may be times when a project can benefit from having several small, related applications within one project. You would take this approach if these small applications are being developed by one team and following the same schedule.

  On the other hand, you might want to create one project to contain common parts such as the data description specifications (DDS) and files, and another project to contain all the source code for your application. You would take this approach if you know that the DDS is going to be used by other applications, or if your application contains a large number of parts.

- **How will the project be named?**

  A project's name has two parts and must be unique on your AS/400 system.

  The project name is what an application developer uses on all the project-related CL commands to identify a particular project. It can be as long as 32 characters. You may, however, want to avoid using the full 32 characters to save typing a very long project name in the control language (CL) command parameters.

  The short project name can be up to 4 characters long and is used to create a unique AS/400 library name for groups in that project. See the *ADTS/400: Application Development Manager/400 User's Guide* for a complete discussion of these naming conventions.

- **Who will be the project's administrator?**

  Project administrators do not require special AS/400 authority to create projects, which means that anyone can create a project. The person who creates a project automatically becomes the project administrator of it. The project administrator defines the groups that make up a project's hierarchy and enrolls other members of the team to the project.

  Every project on your AS/400 system can have a different project administrator, and every project can have more than one administrator. An administrator has update authority to everything in the project, and depending on the nature of the project, it may be advisable for several members of the development team to be authorized as administrators. At the very least, each project should have a designated administrator and a backup.

  Note that in this and future releases, the user profile QSECOFR, which is the system default, is automatically an administrator for all Application Development Manager/400 projects.

- **Will the project administrator also work as an application developer?**

  This is certainly possible and quite likely to occur on small development teams that consist of five or six people. If a project administrator is to act as an application developer regularly, it would be wise for this person to have two user profiles; one profile to be used for project administration tasks, and the other for application development work.

For More Information

When you have answered all the questions this book raises in relation to your development team, its process, and the application to be developed and you are ready to create a project within the Application Development Manager/400 environment, see the *ADTS/400: Application Development Manager/400 User's Guide*. Chapter 2, "Working with a Project" in that book describes how to create a project and work with it using CL commands or the programming development manager utility.

Chapter 5. Planning a Project's Hierarchy

A project hierarchy provides organization and structure to the way a team of application developers work. An Application Development Manager/400 project hierarchy does not impose a process of its own. Hierarchies are created that help development teams to sustain their own processes.

Project hierarchies are never static. They grow and change as the needs of the application change. The Application Development Manager/400 feature makes this growth easy to manage. When required, the project's hierarchy can be expanded to accommodate second and third versions of an application, as well as new groups in which maintenance work on finished versions can take place. Entire projects, including their groups, can be deleted as projects wind down. Some development groups in the project hierarchy can be deleted when the project goes into a maintenance phase.

Project hierarchies can consist of as few as two levels, or as many as 25 levels. You define the number of levels that suits the needs of your development team and the applications they are creating.

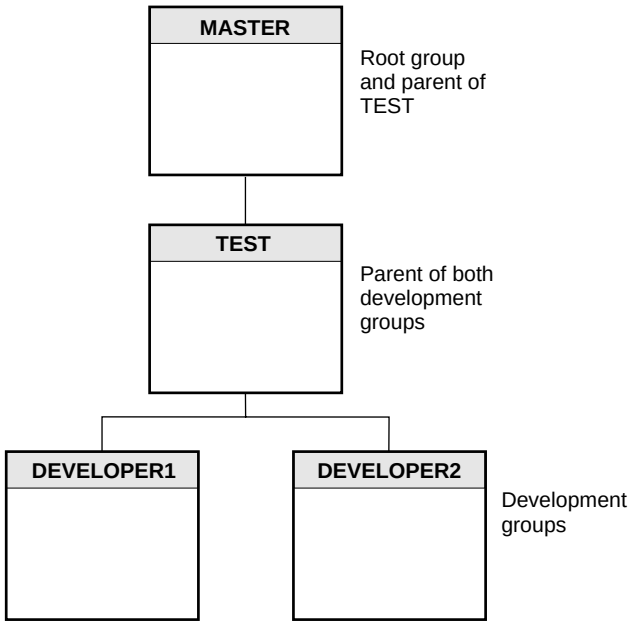

Figure 4. Three levels in the sample project's hierarchy

Figure 4 illustrates a project hierarchy with three levels or phases that manage the development of a single version of a payroll application. These levels represent the phases a typical project goes through while it is being developed, *before* it is ready to export to a production environment. Each box represents a group in the project hierarchy.

The administrator begins to create a project hierarchy by defining the **root group**, which is the first or top group in any hierarchy. The root group in the sample project is called MASTER and will eventually contain the complete version of both the source and objects of the payroll application. There can only ever be one root group in any project hierarchy.

From the root group, the administrator then goes on to define the next stage or level in the hierarchy, in this example, the group TEST. Specifying the group MASTER as its **parent group** creates a hierarchy with the group MASTER directly above the group TEST. A parent group always has groups beneath it in the hierarchy. It is the relationship of a parent group to the groups below it that lets an administrator build a hierarchy one stage or level at a time.

To complete the sample project, the administrator would create the **development groups** DEVELOPER1 and DEVELOPER2 at the bottom or lowest level of the hierarchy. These groups are where application developers typically do the work of developing code. For these groups, TEST is specified as the parent group. As the project administrator defines the groups in this project hierarchy, she or he must also define a **promote code** for the groups. A promote code is the identifier that shows to which group in the project hierarchy a part can eventually be promoted.

For the sample project in Figure 4 the actual work of developing or writing code is done in the groups DEVELOPER1 and DEVELOPER2. Here each developer works in his or her own development group, unaffected by the work being done elsewhere in the project hierarchy. Note that the developer working in the group DEVELOPER1 can see the parts in the group DEVELOPER2 because all developers have read access to all groups in the project hierarchy. However, she or he cannot update those parts.

In this release, the Application Development Manager/400 feature allows you to maintain up to five previous or **archived** versions of each source part that you have changed. If the developer finds that the version that she or he is working on is defective, then the current changes can be undone by **rolling back** to the previous version. When a group is deleted, the archive library associated with it is also deleted. For more information about archiving and rolling back versions, refer to the *ADTS/400: Application Development Manager/400 User's Guide*.

An important feature of change management is keeping track of problems. Such problem tracking is also known as **reason control**. The project administrator can turn the reason control on for any specific group, requiring the developer to specify a part-list part name (**reason**) to record the parts being processed in that group. Whenever parts are created, or changed in a group, or whenever parts are promoted into a group, a part-list (reason) is required to be specified. This will help administrators and auditors to more effectively track changes made due to requirements or enhancements requested.

When a piece of code is complete, the developer moves it to the next stage in the project hierarchy, shown in Figure 4 as the group TEST. This is the first stage where changes from different developers are integrated and tested together. When parts here work as expected, they can be moved to the next level in the project hierarchy. To move one piece of code—or many—from one level to the next implies that a phase in the development process is complete or nearing completion.

When all the components have been integrated and thoroughly tested and the application runs successfully, the appropriate source parts are finally moved to the group MASTER. This is the level where the latest version of the application is normally held. Components of an application held at this level of a project hierarchy may be called production-level code, but they do not become the production version of the application until the application is exported (or copied) from the Application Development Manager/400 environment to a real production environment, as illustrated in Figure 5.

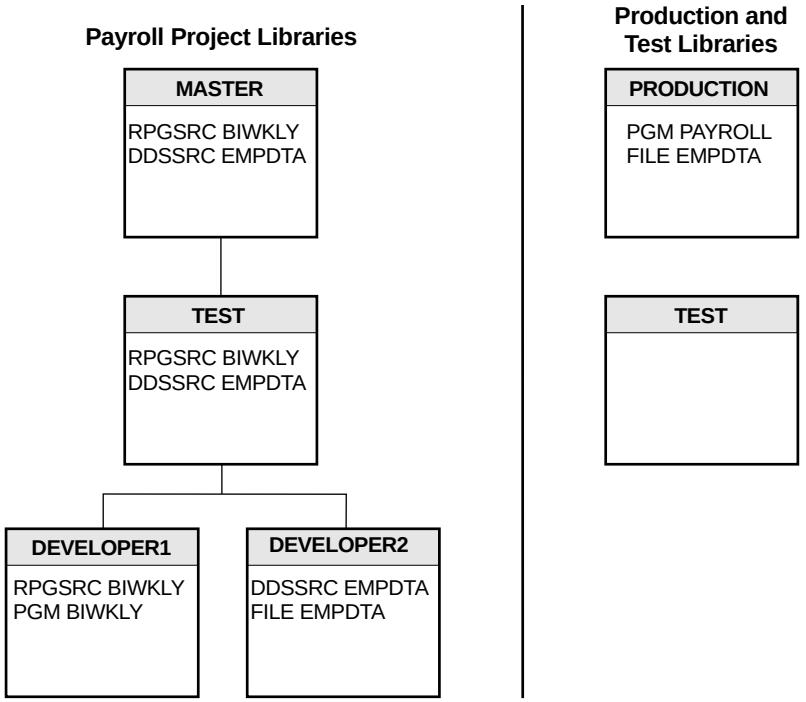

Figure 5. Development and production libraries for the sample project

Questions to Consider

Before you create the groups that make up a project's hierarchy, you must take the time to analyze the application you plan to control and the process your organization follows when developing, changing, and maintaining applications. To define the groups needed for your application, you will need to know the number of phases in your development process, the number of developers working on the application, and the group or groups to which each member of the team should have update access. The following questions will help you to decide on the structure your project hierarchy should have.

- **What phases will your application go through before it is released to production?**

  Each phase of your development process should be represented by a separate level. In Figure 4 three phases are defined: development, test, and completed (or master) phase. Each phase appears as a separate level in the hierarchy. You can define up to 25 levels in any project hierarchy.

Each level in the project hierarchy represents a phase through which all parts should eventually be promoted. If you plan to create many levels in the project hierarchy, remember that someone, that is, the administrator, team leader, or project coordinator, must promote the parts up through all the levels.

- **How many application developers will work on this project?**

  Normally each developer should have her or his own group to work in as illustrated by the groups DEVELOPER1 and DEVELOPER2 in Figure 4 on page 17. You can create as many development groups as you need and your system will hold in your project hierarchy.

- **How will you name your groups?**

  The group name must be unique within the project, which means you must establish a naming convention for all the groups in a project. You may want to consider using the name of the user profile of each developer or tester for whom you are creating a group. If your project hierarchy contains several versions of one application, you may want to start each group that belongs to one version with a common prefix. For example, the names of all the groups for the first version of your application could begin with V1.

  As you create each group, you must gives it two names.

  – The group name up to 32 characters long that an application developer can then use on all the group-related CL commands to identify a particular group. You may, however, want to avoid using the full 32 characters to save typing very long project names in the CL command parameters.

  – The short group name up to 5 characters long is used by the Application Development Manager/400 feature in combination with the short project name to create a unique AS/400 library name representing a specific group.

  For example, in the sample project called PAYROLL with a short name of PAY, there is a group called MASTER with a short name of MST. The name of the AS/400 library for this group would be PAY.MST. When the test group is created in the same project, with the name TEST and a short name of TST, the library name for this would be PAY.TST.

  See the *ADTS/400: Application Development Manager/400 User's Guide* for a complete discussion of these naming conventions and the rules to remember as you create names for projects and groups.

- **Which groups do you want to have archived versions?**

  You can archive parts in any groups. You may want to consider archiving parts in the group where the version of the code is being tested or ready to ship, and you may need to roll back the previous version of the code, if the need arises. Every time you change or promote a part with ARCHIVE(*YES), a copy of that part is automatically stored in an archived library. Up to five versions of each source parts recently changed or promoted will be stored. An archived version can be retrieved or rolled back using the IMPPART command.

- **Will the reason control function be activated?**

  If PARTLREQ(*YES) is specified when creating or changing a group, developers issuing part commands in that group must specify a valid part-list part name on the command. This gives the administrators the ability to ensure all changes are being tracked to a given requirement or enhancement (reason).

- **Who will verify that all the components in the application work together?**

  This could be the team leader or a group of people who test the application. If you have a test team, you may want to create a separate group in the hierarchy for each tester.

- **How many versions of the application will you support at one time?**

  For example, do you intend to maintain a copy of the current application, and at the same time, update or add new function to the same application? In this scenario, you would create groups for the current version and groups for the new version you are updating or adding to. Each version should have a unique promote code to prevent developers from promoting parts from one version into the groups of the other version.

- **If you are supporting several versions of one application, which developers will update the code in each version?**

  If the same developer or group of developers updates both versions, each person should have a separate development group for each version.

- **Will your application support more than one language for its messages, display files, or print files?**

  If your application is developed in one language, for example, English, but the messages, reports, and displays are also available in other languages, you can create a different branch in the hierarchy for each of the languages you support.

For More Information

When you are ready to create your project and its groups, see the *ADTS/400: Application Development Manager/400 User's Guide*. Chapter 3, "Working with Groups in a Project Hierarchy" in that book describes how to create groups using CL commands or the programming development manager utility.

Chapter 6. Planning Enrollment

Using the Application Development Manager/400 feature, a development team works in an environment that controls where individual members of the team can make changes to pieces of code. In the sample project, the groups DEVELOPER1 and DEVELOPER2 have been set up for two different developers. With the typical enrollment procedure, each of these developers has update access to a specific group and, at the same time, read access to the other groups in the project. Data security is ensured because developers can only change parts in the groups to which they have update access.

Specific members of a development team may also have update access to higher groups, according to the needs of the project and the process the development team is following. For example, the project leader might be the person with update access to the groups TEST and MASTER. This access means that a project leader will be able to promote parts up through all the levels of the hierarchy and build the application in any one of these groups as the code moves through the development cycle. The member of the team who acted as administrator has update access to *all* the groups in the project hierarchy.

How you enroll the members of your development team, and what access they have to which groups, depends on the needs of your application and the processes your team plans to use. The following section discusses some of the topics you should consider before you start to enroll people to the project you have created.

Questions to Consider

The following questions will help you decide how the various members of the development team should be enrolled to the project.

- **What security requirements does your organization have?**

  All users enrolled to a project can read all the information stored in it. You cannot restrict read access within a project. No special OS/400 authority is required to enroll a user as a developer or as an administrator. Note that the user profile QSECOFR is also automatically made an administrator for all Application Development Manager/400 projects.

- **What types of access will different members of the development team need?**

  You must determine who needs update access to the entire project hierarchy, to a few groups in the project hierarchy, to one group in the project hierarchy, or no update access at all (strictly read access). For example, a project coordinator or team leader may require update access to the entire project or to several development groups. A developer may require update access to only one development group. A person responsible for testing may not require update access to any groups.

- **Will each developer have his or her own group?**

  If your application has a high rate of change activity, it is probably best to insulate each developer's work from another's in the early stages of development. Applications undergoing a high rate of change can create an unstable environment for developers who are sharing a development group.

- **Will developers share development groups?**

  Shared development groups are groups where more than one developer does part development activities at the same time. This can be advantageous when the change activity in an application is low. It means that the number of steps required to move a part up the project hierarchy can be reduced, and interaction between developers is made easier. By sharing a development group, developers working on the same enhancement in related parts do not need to promote the parts or use a special search path to compile and test their parts together. A **search path** in any project hierarchy is an arrangement of groups in the order they are to be searched when looking for parts. Search paths through a project's hierarchy are there to help application developers to work more productively.

  If you decide that developers will share a development group, the Application Development Manager/400 feature ensures the integrity of their code. When a developer checks a part out to a development group, the part is locked to that developer's user profile. No other developer has access to the part. An **access key** identifies the user profile of the developer who is currently working with a part.

- **Who will promote parts to the target group?**

  A **target group** is the highest group in the project hierarchy with the same promote code to which a part can be promoted. If the person promoting parts to this group is the project administrator, he or she has update access to all groups automatically. If the person is an application developer, he or she requires update access to all the groups between the development group and the target group. A developer can promote parts to the group in the hierarchy *above* the group to which he or she has update access.

- **Who will compile parts in each group in the project?**

  The person who is compiling parts must have update access to the group in which he or she is working. If this person is the project administrator, he or she has update access to all groups in the project hierarchy. The administrator can compile one or two parts in a development group or the entire application based on the parts held in the master group. If this person is an application developer, he or she can only compile parts in the group to which he or she has update access.

For More Information

If you are developing an entirely new application for which no code exists, your planning is complete. You are ready to begin the work of creating a project, defining its groups, and enrolling the application developers. See Chapter 2, “Working with a Project” and Chapter 3, “Working with Groups in a Project Hierarchy” in the *ADTS/400: Application Development Manager/400 User’s Guide*.

If you intend to work with an application that already exists, you must import it into the Application Development Manager/400 environment. Continue reading this book. Chapter 7, “Planning to Import an Application” discusses what importing means and the planning questions you should consider.

Chapter 7. Planning to Import an Application

When a development team is starting an Application Development Manager/400 project and bringing an existing application into a project hierarchy, the necessary source code and objects of the application have to be **imported** to be under the control of this feature. Importing an application copies it from an AS/400 library into the Application Development Manager/400 environment. The person doing the importing must have update access to the group to which the application is being imported.

Parts can be imported into any group in the project hierarchy. Which group depends on who is doing the importing, the group to which the importer has update access, how many parts are being imported, and the purpose for which they are being imported.

If you are importing a complete application, it is best to import it into a group at a higher level in the hierarchy, such as the master group. Or, you may prefer to define a group specifically to hold the imported components as illustrated in Figure 6. Once imported, the application should be compiled and tested to ensure that it works as expected and that all the parts that should have been imported were copied correctly. (Chapter 8, “Planning to Build an Application” describes the Application Development Manager/400 build process and discusses the issues you should consider before you compile and test your application.) Only when you are sure that the application, as imported, works as expected can the development team begin the work of enhancing the application.

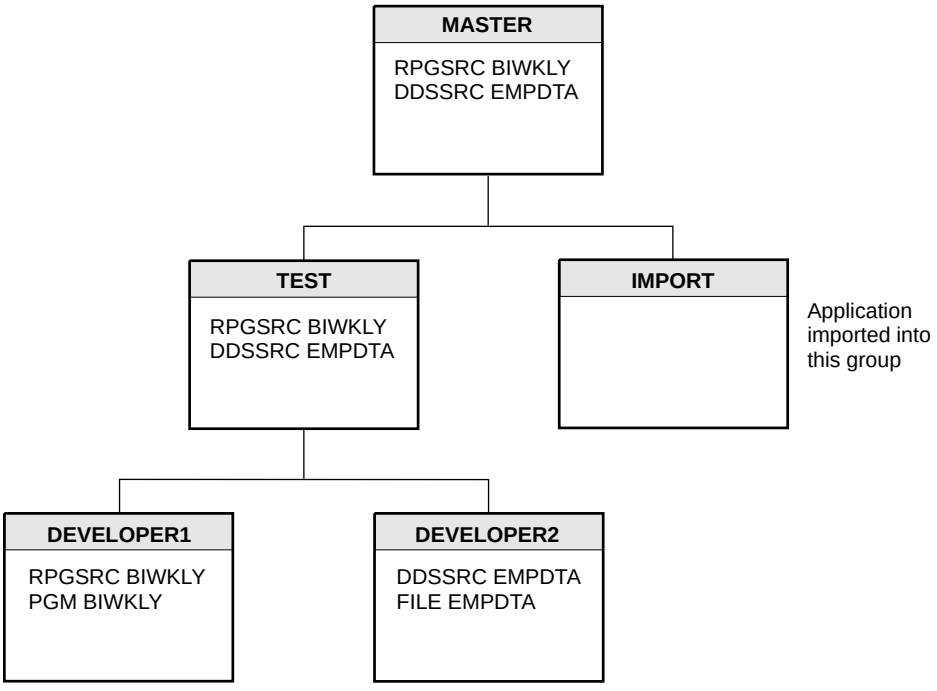

Figure 6. Importing parts into a new group in the project hierarchy

Questions to Consider

The following questions will help you to develop a strategy for importing an application. To import an application in the Application Development Manager/400 environment, you must first create a project.

- **What type of import strategy will you use?**

  You can choose to import an entire application at once or in stages. You can choose to import only those objects that are to change. The important thing to remember is that the Application Development Manager/400 feature is most effective when as many components as possible are brought under its control. Doing so produces the maximum benefits from the build process.

- **Into which group in the project hierarchy will you import the application?**

  If you are importing an application into an empty project hierarchy, consider importing all the parts into the master group. If you import the parts into a development group, eventually someone will have to promote all the parts up through several levels of the hierarchy to the target group.

- **What files and objects will make up your application?**

  You need to identify all the components of your application, and then determine if each object and file is supported as an Application Development Manager/400 part type. If your application uses AS/400 objects or source members that are not supported by the Application Development Manager/400 feature, you can define your own part types for these components. These are called user-defined types which you define so that they are recognized by the Application Development Manager/400 feature.

  See the *ADTS/400: Application Development Manager/400 User's Guide* for a list of all the AS/400 files and objects that can be imported into a project hierarchy. See "Planning for a Mixed Application" on page 28 for information on how to manage application development in a mixed environment.

- **How many items will you import?**

  The amount of time it takes to import an application depends on the number of items to be imported. The more objects, source files, and source members there are in your application, the longer the import operation takes. For most applications, the import operation will not take very long. You can submit the Import Part (IMPPART) command in batch while you work on something else. Each item to be copied is treated as separate import operation.

  If your application has many logical files, the import operation will take longer. You must import the physical files on which these logical files are built first.

- **Will you import the application into a project hierarchy that already contains parts?**

  If you are working on a new release of an application you may want to import only the parts to be changed into the project hierarchy. To do this, consider creating a new group whose parent is the root group in the project hierarchy as illustrated in Figure 6 on page 25. This group will prevent any disruption of parts being developed and promoted in other branches of the project hierarchy.

- **How will you test the parts you have imported to ensure you have imported everything the application needs?**

  You should consider creating a special search path to use when you build the parts you have imported. This search path should consist of only the group IMPORT. Building the application this way will ensure that all the parts you imported work together as expected before you promote them to the group MASTER, and before you build them with parts in other groups.

- **Will you use the programming development manager utility to import several AS/400 objects and source file members at one time?**

  This utility provides two system-supplied user-defined options: IO to import AS/400 objects into a project hierarchy, IM to help you to import source members. These options can be found in the system-supplied options file QAUOUSR in the library QPDA. Use these options with F13=Repeat on any programming development manager utility menu to import many objects or source members at one time. This is useful if you want to import a subset of application components that cannot be done with one or two IMPPART commands.

  Use the IO user-defined option from the Work with Objects Using PDM display.

  ```
  IO  IMPPART OBJ(&L/&N) OBJTYPE(&T) TYPE(&S) PART(&N) TEXT(&X)
  ```

  Use the IM user-defined option from the Work with Members Using PDM display.

  ```
  IM  IMPPART OBJ(&L/&F) OBJTYPE(*FILE) MBR(&N) PART(&N) LANG(&S) TEXT(&X)
  ```

  For more information about these system-supplied options, see the section entitled “Working with User-Defined Options” in Chapter 15 of the *ADTS/400: Application Development Manager/400 User’s Guide*.

- **Will you import only source members and not objects that can be compiled?**

  This is the recommended practice for high-level-language programs as well as data description specifications (DDS). Be sure to build your application after you import all the pieces to verify that the application works as expected and to establish the relationships between the parts of the application.

  If you do import the *PGM, *CLD, *CMD, and *FILE objects and build the application using the parameters recommended for the Build Part (BLDPART) command, all the objects for which the source was also imported are recreated in any case.

- **Will you import records longer than 92 characters?**

  Files created by the Application Development Manager/400 feature to store imported source members have a record length that is determined by the part type.

- **Will you import pieces of code to be used by several applications into a separate project?**

  High-level-language programs or DDS that fall into this category are candidates for a shared project.

- **Will you have to import the standard high-level-language include files or system includes that your application uses?**

  For example, you do not need to import the information stored in the system include members for the C/400 language that is stored in the source physical file H.

- **Will you import the data that is associated with your physical files?**

  You can copy the data from the database members associated with physical files by specifying DATA(*YES) on the IMPPART command. For example, if the data is reference data that you can use while you test your application, you may want to copy it to save time later in the testing phase of your development cycle. However, if you need to create specific data for test purposes, it is probably not worthwhile copying the data.

- **Will your development team use special compiler options?**

  When working with the Application Development Manager/400 feature, developers use the Build Part (BLDPART) command to compile or process the parts in an application. You specify the compile options you need for your application by creating a build options part (part of type BLDOPT) with the same name as the part to be compiled.

- **Will the files you import be journaled?**

  Imported journaled files are not journaled in the project hierarchy because the Application Development Manager/400 feature does not support journals and journal receivers as parts. You must create the journal and journal receiver and then attach the file to the journal for these imported files to be journaled.

Planning for a Mixed Application

A **mixed application** is an application that has some components under Application Development Manager/400 control and others that are not. This also requires some planning because in a mixed application components can be stored in the several places.

- **Which components will be held in the project hierarchy?**

  The project hierarchy contains components of the application that are supported as part types by the Application Development Manager/400 feature. You should move all the components of your application that are supported by this feature to a group in a project hierarchy.

  The more information that is within Application Development Manager/400 control, the better. Warnings will be issued during the build process if components that exist outside the Application Development Manager/400 environment are referred to. This is because the build process cannot recognize the relationships between parts that are under Application Development Manager/400 control and parts that are not.

- **Which components will be in a test library?**

  A test library contains all those components of your application that are not under Application Development Manager/400 control.

- **Which components will be in the library QSYS?**

  If your application requires system objects such as special user profiles or device descriptions, these reside in the QSYS library. Special system objects cannot be stored in test libraries.

  Objects contained in QSYS cannot have multiple versions. If you have a requirement for more than one version of such an object, you can do this indirectly by storing the CL source that creates it in an Application Development Manager/400 part.

- **Which components will be in a production library?**

  A production library contains your real data and the program objects (*PGM) for your application.

- **Which components of your application will be held in a different project hierarchy?**

  If you are importing components that are to be shared across different projects, you may want to import them into their own project. From here you can access them as you need them by defining alternate search paths.

For More Information

When you are ready to import an application, see the *ADTS/400: Application Development Manager/400 User's Guide*. Chapter 6, "Importing an Application" in that book describes how to import both complete applications and just one or two parts.

Chapter 8. Planning to Build an Application

When the time comes to compile complete applications or individual parts, the development team can rely on the power and flexibility the build process provides. Working in the Application Development Manager/400 environment, developers now only have to indicate which parts they want to be built. They no longer have to analyze the relationships between these parts or define files describing these relationships. The build process does this for them, thereby freeing them to do other work. For the development team, this means great savings in time and effort throughout the development cycle and ensures there are no level checks of the application during run time.

The power of the build process comes from its ability to analyze what needs to be compiled and what does not. Using this process, developers enrolled in a specific project can build one part at a time or many parts. Whether they are building one part or many, they know that the integrity of their data is assured. The build process determines which parts of an application have changed and, based on the relationships between those parts, recompiles them so that all parts are current in relation to their **dependencies**. (A dependency is the relationship between two parts where one part requires another part in order to be built.) Because the build process analyzes and maintains the relationships between parts, the Application Development Manager/400 feature is most effective when as many parts of an application as possible are under its control. In this way, the development team derives the maximum benefits from the build process.

The build process is also extremely flexible. Developers can build a single part or an entire application. A development team can establish the guidelines that describe how parts are to be built and what is required before they can be promoted to the next level in the project hierarchy. Build options (parts of type BLDOPT) can be created that establish common compiler options so that the build process can be run again and again, and parts are built the same way every time. User-defined options can be created in the programming development manager utility that further ensure the team builds parts in a consistent manner. Search paths can be defined to simplify the building and testing of parts across the life of the application and across more than one project. This versatility, combined with the power of the build process, produces significant productivity improvements for any application development team.

The Build Process

Let's return to the sample project to see how the build process looks at parts, what happens when parts are compiled, and the objects (or files) that are created.

In Figure 7, a developer has checked the parts RPGSRC BIWKLY and DDSSRC EMPDTA out to the group DEVELOPER1 to make some changes. These changes can be something as small as a changed field.

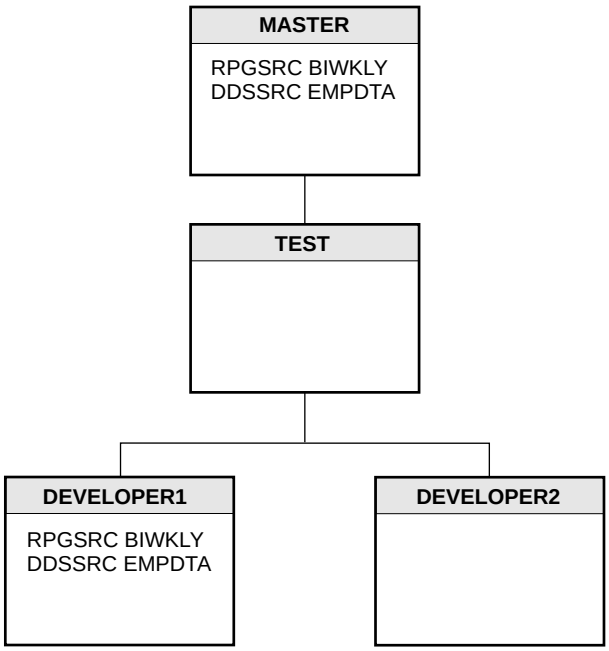

Figure 7. Parts checked out to the group DEVELOPER1

These parts have been changed, and now must be compiled to make sure they work as expected. Using the BLDPART command, the developer specifies that the parts RPGSRC BIWKLY and DDSSRC EMPDTA are to be built in the group DEVELOPER1.

When build processing is complete, the group DEVELOPER1 contains four parts, the two original parts, RPGSRC BIWKLY and DDSSRC EMPDTA, and two new ones, PGM BIWKLY and FILE EMPDTA as shown in Figure 8.

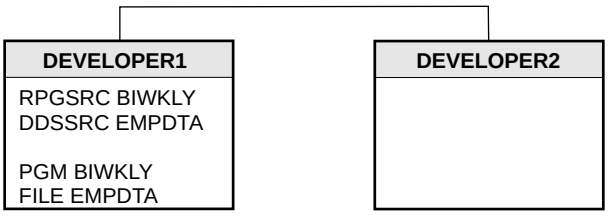

Figure 8. Parts built in the group DEVELOPER1

The original parts are now **current**, which means that they have been built with the latest version of all the source and related parts used to create them. In contrast, a part is **stale** if the source and related parts have changed since the part was last built. The build process understands when parts need to be built because it recognizes when parts are current and when they are stale.

When it is clear that the parts RPGSRC BIWKLY and RPGSRC MNTHLY work as expected in the group DEVELOPER1, they can be promoted to the next level up the hierarchy. During the early phase of the development cycle, tested, stable parts are often promoted to a higher group where they are readily available to be compiled and built with parts being developed by other members of the development team. In Figure 9, the parts have been promoted to the test group.

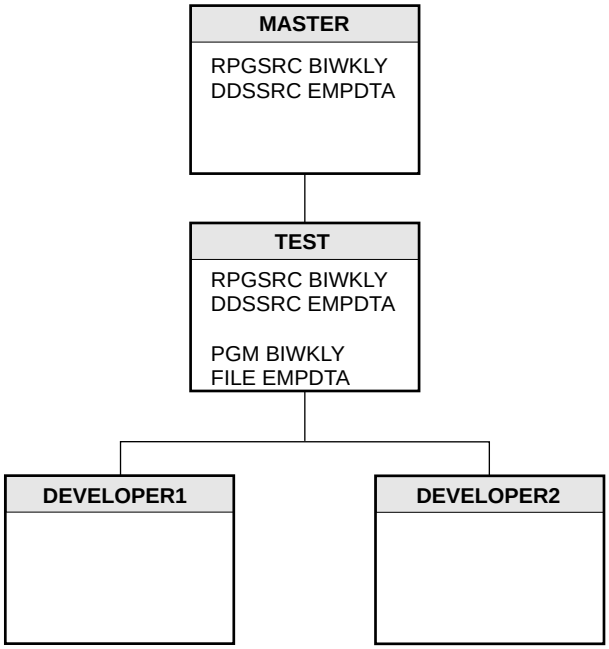

Figure 9. Parts promoted to the group TEST

Now, if a developer working in the group DEVELOPER2 builds another part that depends on either RPGSRC BIWKLY or DDSSRC EMPDTA, the build process, using the default search path, finds the most current version of the part in the TEST group. The parts are built, and in this instance the build outputs are stored in DEVELOPER2.

The above example has described what happens when one or two parts are built in a developer's own group. Much the same thing happens when build processing is done in groups higher up the project hierarchy. The differences involved in building parts at higher levels have to do with the process the development team is following, the needs of the application, which members of the team have update access to these other groups, and where in the development cycle the project is. To be able to build parts in these groups, a developer must have update access to the group. At the test level, the person building parts might be a tester or the test coordinator.

Build scope

Building parts at higher levels may also mean that the scope of build processing has expanded. Generally, a developer building a part in a development group would choose the **limited** scope, to build only the specified part. This is typically how developers build parts early in the development cycle.

When developers are confident that parts work as expected, they can expand the build scope to build several or many parts. When the **normal** scope is used, the specified part and all the parts on which it depends and on which they depend, and so on, are built. The **extended** build scope is much the same as the normal build scope, but it takes the build processing one step further, in that any parts related to all the parts being built are also built. The normal and extended build scopes ensure that all dependent parts are built and guarantee no more level checks.

Build options

The build command uses the default values associated with each optional parameter on the compiler commands when parts are processed. These values can be changed by creating a new build options part. This part allows a developer or team of developers to specify the compiler or processing commands needed for their particular project, group, or part.

Search paths

Search paths add flexibility and versatility to the way a development team uses the power of the build process. A search path is an arrangement of groups that determines the order in which the Application Development Manager/400 feature searches the various groups when looking for parts in a project hierarchy. While the build process looks at dependencies between parts, it normally uses only the default search path. In the sample project, the default search path gives the developers working in the groups DEVELOPER1 and DEVELOPER2 access to the parts held in any of the groups above them in the project hierarchy.

Developers can, however, also create **alternate search paths** that give them different views of a project hierarchy. They can create a search path that excludes groups in the search path, includes groups not in the search path, or adds groups from a different project to the search path. This is done by creating a search path part that lists the groups the build process is to use as it builds the parts they specify on the BLDPART command. For example, as the project nears completion, the team leader might want a search path that excludes parts held at the development and test levels. To do this, a search path part would be defined that only looks at current parts in the MASTER group.

One such alternate search path is a **cross-project search path** that contains groups from other projects so that common information is shared across projects, such as a file that contains data about an employee that is used by both a payroll application and a personnel application. The cross-project search path helps to maximize the initial investment in developing the application, and saves the development team the time and effort of redeveloping code that already exists or of maintaining multiple copies of that code.

The build report

A build report is always generated whenever an application is built. This report describes the results of a build process. Developers can view this report on a display, or they can print it. It provides the following information:

- The parameters and values used on the BLDPART command
- The search path used (groups in this path are listed in the search path part)
- The search path part used
- The bind-step options specified
- A list of warning and error messages issued during the build process
- The name of the compiled source part and the group where it was found
- The type of output that was created, such as a program or file
- The name of the output part that was created
- The reason the part was built
- The name of the BLDOPT part used, if any, and the group where it was found.

Questions to Consider

You will need to develop a build strategy for every project you create. How you build an imported application may well be different from the build method you would use for a new application still being developed. You should build an application you have imported to verify that all the parts required have been imported successfully, and to develop a record of the relationships that exist between the parts, as imported. Before you build your application, you should have answers to the following questions.

- **Who will build parts and in which groups?**

  Whoever uses the BLDPART command must have update access to the group where build processing is to take place. Typically, this means that a developer can only build parts in the development group to which he or she has access. But someone with administrator access to the project can build parts at all the levels in the project hierarchy. If your organization wants a team leader or project coordinator to build parts in higher levels than the development groups' level, those individuals must have update access to those groups.

  Once you have decided who builds parts at the various levels or groups in the project hierarchy, there are several approaches that you can take:

  – Developers can tell the team leader or administrator when their work is complete and promoted to the next group. The administrator then builds the appropriate parts in that higher group.

  – The team leader or administrator can establish a routine for building parts. A build operation could be performed once a week on a certain day when development activity is high. In periods of less activity, a build operation could be performed biweekly or monthly.

- **Where will parts be built?**

  In the sample project, the developers working in the two development groups build parts in each development group and then promote these parts to the group TEST. The project leader may be the person who builds parts here. A final build operation is performed in the group MASTER.

- **When will parts be promoted?**

  In the sample project, the project leader must decide when it is necessary to promote parts from the group TEST to the group MASTER. This may happen once or twice a week, at a specified time, after which all parts at the master level are to be built. The thing to remember is that parts used in the build operation are found based on the group specified in the BLDPART command. Parts are found in that group, and in groups higher in the search path. Groups below the group specified on the BLDPART command are not used to find parts. If all parts have not been promoted to the master group, the most current version of the application is not built when the master group is specified on the BLDPART command.

- **Will the development team use only the compiler defaults when processing programs?**

  If this is the case, you need to create parts of type BLDOPT only to build parts of type PGM using the CRTPGM command, and CMDSRC in your application.

- **Will the development team use compiler options consistently in groups at each level of the hierarchy?**

  In development groups, you could, for example, create parts of type BLDOPT. that ensure programs are compiled with debug information, generate compiler listings, and include second-level message text in the compiler listing. In test groups, you could create parts of type BLDOPT that ensure programs are compiled without debug information and generate compiler listings. In the root group, you could create a part of type BLDOPT to ensure that programs are compiled without debug information, without generating a compiler listing, and that the programs are optimized.

- **Will the development team use the default compile options?**

  The build process searches for a BLDOPT part with a name that matches the name of the part being built. If such a part is not found, the build process searches for the part BLDOPT QDFT. This is the default build options part. See the *ADTS/400: Application Development Manager/400 User's Guide* for an explanation of how to use a build options part with this name.

- **Will the development team compile certain components or individual programs within applications in a consistent way?**

  Some programs that are very large and complex, for example, could be optimized when compiled in development groups.

Testing an Application

When you import an application into the Application Development Manager/400 environment, it should be tested to verify that all the parts that need to be imported have in fact been imported and that the application works as expected. Testing your application allows you to preserve the source and objects that you exported to a test library. This, in turn, helps you to debug the application because you know precisely which source created which objects in the test library.

The test environment within the Application Development Manager/400 feature is meant to be similar, but cannot be identical, to the actual production environment. The following conditions may affect how successful your testing procedures are.

- **Will your application update database files?**

  If your application does, the person testing the application may not necessarily have the appropriate authorization to those files. The database files must reside in the developer's or tester's group. In other words, the developer or tester must have update access to the group that contains the database files.

- **Will your application include library-qualified files or program calls?**

  Parts used within a project hierarchy must be found using the search path defined by the project hierarchy or defined in a part of type SCHPTH. If files or objects that are referenced within parts are library-qualified, such as for example, CPROG/HELLO rather than *LIBL/HELLO, the file or object that is referenced might not be found in the search path that is used when the application is built and run.

- **Will your application include file overrides?**

  If you attempt to override files, or if file overrides are incorporated into parts, unpredictable results can occur. Developers should avoid overriding files in parts, and developers and testers should not attempt to override files when testing the application in the Application Development Manager/400 project hierarchy.

- **Will you be testing other AS/400 system objects not directly supported by the Application Development Manager/400 feature and for which you have not defined any user-defined types?**

  Applications that use objects or file types not supported by the Application Development Manager/400 feature can still be tested from within its control. You can add the AS/400 libraries that contain the files and objects required for the application to the library list. Use the CL command Add Library List Entry, (ADDLIBLE) to do this. This command can be used before or after the project libraries are added to the library list using the ADDPRJLIBL command.

For More Information

When you are ready to build an application, see the *ADTS/400: Application Development Manager/400 User's Guide*. Chapter 11, "Building an Application" in that book describes how to build complete applications and just one or two parts.

Chapter 9. What to Do Next

When you have considered all the questions posed by this book, you are ready to begin the task of creating your own application development environment. Please use the following worksheets to record the needs of your development environment. These worksheets repeat the questions found in the relevant chapters in this book and provide you with space to make notes about your development environment, the members of the development team, and the processes they follow.

For more information about the steps outlined briefly in Chapter 3, “Planning the Application Development Environment,” see the *ADTS/400: Application Development Manager/400 User’s Guide*. This book describes these steps in detail and provides you with many examples of the actual commands you will be using. Working with the information you record on these worksheets and the examples in the *ADTS/400: Application Development Manager/400 User’s Guide*, you can create an application development environment under the control of the Application Development Manager/400 feature quickly and easily.

The information you need in the *ADTS/400: Application Development Manager/400 User’s Guide* is as follows:

- Chapter 2, “Working with a Project”

  This chapter explains how to use the Create Project (CRTPRJ) command and describes several other project-related commands.

- Chapter 3, “Working with Groups in a Project Hierarchy”

  This chapter explains how to use the Create Group (CRTGRP) and Add Project User (ADDPRJUSR) commands and discusses promoted codes, notification, how to work with groups. It also includes an example that takes you through the steps to create a project hierarchy similar to the payroll application used in this book.

- Chapter 6, “Importing an Application”

  This chapter explains how to use the Import Part (IMPPART) command and provides an example that takes you through the steps to import an application.

- Chapter 11, “Building an Application”

  This chapter explains how to use the Build Part (BLDPART) command and includes an example that takes you through the steps to build an application.

The *ADTS/400: Application Development Manager/400 User’s Guide* contains much more information, of interest to both application developers and project administrators. In it are descriptions of how to work with parts, how to test an application, and what to do when you are ready to export you application to a production environment. You will also find detailed information about search paths, promote codes, user-defined types, security information, and how to work with the Application Development Manager/400 feature using the programming development manager utility.

Planning Worksheet 1: Planning a Project

This worksheet repeats the questions discussed in Chapter 4.

1. **What will the project contain?**

2. **How will the project be named?**

3. **Who will be the project's administrator?**

4. **Will the project administrator also work as an application developer?**

Planning Worksheet 2: Planning a Project's Hierarchy

This worksheet repeats the questions discussed in Chapter 5.

1. **What phases will your application go through before it is released to production?**

2. **How many application developers will work on this project?**

3. **How will you name your groups?**

4. **Which groups do you want to have archived versions?**

5. **Will the reason control function be activated?**

6. **Who will verify that all the components in the application work together?**

7. **How many versions of the application will you support at one time?**

8. **If you are supporting several versions of one application, which developers will update the code in each version?**

9. **Will your application support more than one language for its messages, display files, or print files?**

Planning Worksheet 3: Planning for Enrollment

This worksheet repeats the questions discussed in Chapter 6.

1. **What security requirements does your organization have?**

2. **What types of access will different members of the development team need?**

3. **Will each developer have his or her own group?**

4. **Will developers share development groups?**

5. **Who will promote parts to the target group?**

6. **Who will compile parts in each group in the project?**

Planning Worksheet 4: Planning to Import an Application

This worksheet repeats the questions discussed in Chapter 7.

1. **What type of import strategy will you use?**

2. **Into which group in the project hierarchy will you import the application?**

3. **What files and objects will make up your application?**

4. **How many items will you import?**

5. **Will you import the application into a project hierarchy that already contains parts?**

6. **How will you test the parts you have imported to ensure you have imported everything the application needs?**

7. **Will you use the programming development manager utility to import several AS/400 objects and source file members at one time?**

8. **Will you import only source members and not objects that can be compiled?**

9. **Will you import records longer than 92 characters?**

10. **Will you import pieces of code to be used by several applications into a separate project?**

11. **Will you import the standard high-level-language include files or system includes that your application uses?**

12. **Will you import the data that is associated with your physical files?**

13. **Will your development team use special compiler options?**

14. **Will the files you import be journaled?**

15. **Which components will be held in the project hierarchy?**

16. **Which components will be in a test library?**

17. **Which components will be in the library QSYS?**

18. **Which components will be in a production library?**

19. **Which components of your application will be held in a different project hierarchy?**

Planning Worksheet 5: Planning to Build an Application

This worksheet repeats the questions discussed in Chapter 8.

1. **Who will build parts and in which groups?**

2. **Where will parts be built?**

3. **When will parts be promoted?**

4. **Will the development team use only the compiler defaults when processing programs?**

5. **Will the development team use compiler options consistently in groups at each level of the hierarchy?**

6. **Will the development team use the default compile options?**

7. **Will the development team compile certain components or individual programs within applications in a consistent way?**

8. **Will your application update database files?**

9. **Will your application include library-qualified files or program calls?**

10. **Will your application include file overrides?**

11. **Will you be testing other AS/400 system objects not directly supported by the Application Development Manager/400 feature and for which you have not defined any user-defined types?**

Bibliography

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

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

- *ADTS/400: Application Development Manager/400 API Reference*, SC09-1809
- *ADTS/400: Application Development Manager/400 Introduction and Planning Guide*, GC09-1807
- *ADTS/400: Application Development Manager/400 User's Guide*, SC09-1808

Besides the printed library, the Application Development Manager/400 feature also offers a demonstration diskette that will introduce developers to this application development environment.

- *ADTS/400: Application Development Manager Demo Diskette*, GV40-0828

The Application Development ToolSet/400 library contains 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 Application Development ToolSet/400 and the AS/400 Server Access Programs*, SC09-1939

The Application Dictionary Services/400 library contains the following publications:

- *ADTS/400: Application Dictionary Services/400 Self-Study Guide*, SC09-1904
- *ADTS/400: Application Dictionary Services/400 User's Guide*, SC09-1860

The CODE/400 library contains the following publications:

- *CODE/400 Debug Tool*, SC09-1905
- *CODE/400 General Information*, GC09-1907
- *CODE/400 Keyboard Template*, GX09-1297
- *CODE/400 Quick Reference Card*, GX09-1296
- *CODE/400 Self-Study Guide*, SC09-1911
- *CODE/400 Installation*, SC09-1908
- *LPEX Command Reference*, SC09-1910

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

- *Backup and Recovery – Basic*, SC41-3304
- *CL Programming*, SC41-3721
- *CL Reference*, SC41-3722
- *COBOL/400 User's Guide*, SC09-1812
- *COBOL/400 Reference*, SC09-1813
- *Data Management*, SC41-3710
- *DB2/400 Database Programming*, SC41-3701
- *DB2/400 SQL Programming*, SC41-3611
- *DB2/400 SQL Reference*, SC41-3612
- *DDS Reference*, SC41-3712
- *Experience RPG IV*, SC09-1938
- *ICF Programming*, SC41-3442
- *ILE Application Development Example*, SC41-3602
- *ILE Concepts*, SC41-3606
- *ILE C/400 Programmer's Guide*, SC09-1820
- *ILE C/400 Programmer's Reference*, SC09-1821
- *ILE C/400 Reference Summary*, SX09-1288
- *ILE COBOL/400 Programmer's Guide*, SC09-1522
- *ILE COBOL/400 Reference*, SC09-1523
- *ILE COBOL/400 Reference Summary*, SX09-1260
- *ILE RPG/400 Programmer's Guide*, SC09-1525
- *ILE RPG/400 Reference*, SC09-1526
- *ILE RPG Reference Summary*, SX09-1261
- *International Application Development*, SC41-3603
- *National Language Support*, SC41-3101
- *Publications Ordering*, SC41-3000
- *Publications Reference*, SC41-3003

- *Security – Reference*, SC41-3302
- *System API Programming*, SC41-3800
- *System API Reference*, SC41-3801
- *System Manager/400 Use*, SC41-3321
- *System Operation for New Users*, SC41-3200

Index

E

G

H

I

J

L

M

N

O

P

R

S

T

U

V