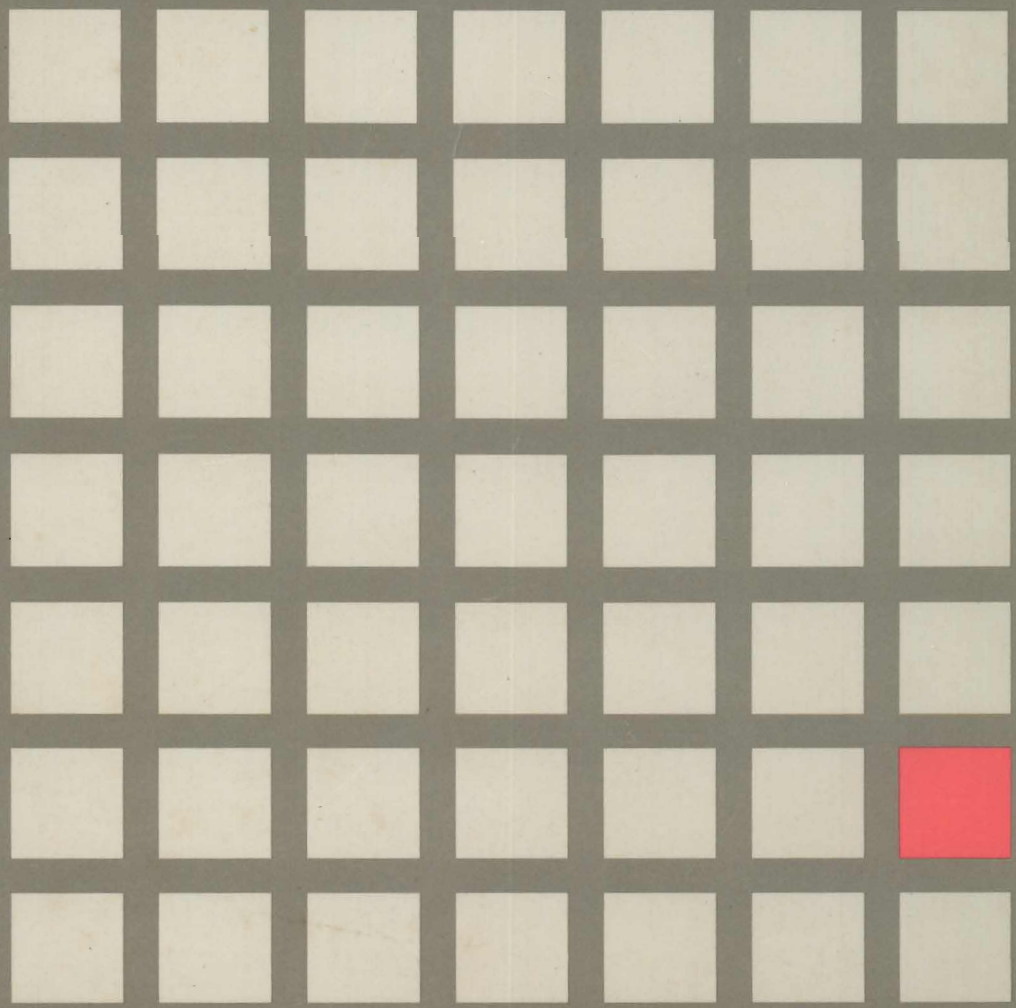
IBM

VM/Integrated System

SC24-5305-01

Developing An Application : Getting Started

5.1



VM/Integrated System

SC24-5305-01

Developing An Application : Getting Started

5.1

Second Edition (May 1988)

This edition, SC24-5305-01, applies to the VM/Integrated System that is based on Release 5.1 of VM/Integrated System BASE (Program 5664-301). This edition applies to all subsequent releases until otherwise indicated in new editions. Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a list of changes, see "Summary of Changes" on page 115.

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 program may be used instead.

In this manual are illustrations in which names are used. These names are fanciful and fictitious, created by the author; they are used solely for illustrative purposes and not for identification of any person or company.

Ordering Publications

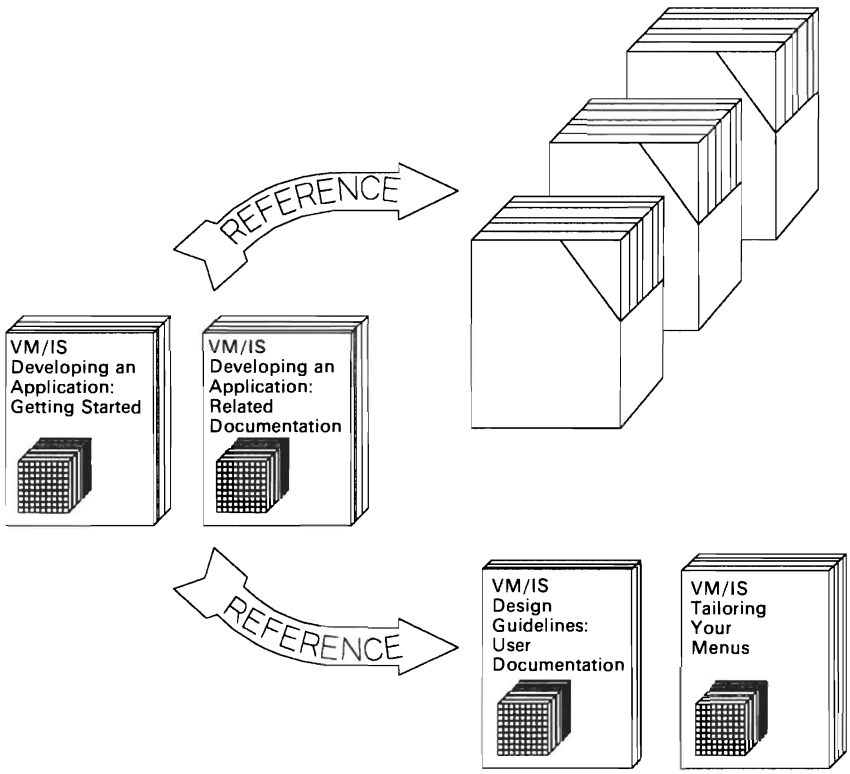
Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to: IBM Corporation, Information Development, Dept. G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

About The Developing an Application Series

To help you create applications for the VM/Integrated System (VM/IS), IBM has many publications that contain application development information about the products and functions of VM/IS. These publications contain the information you need to help you plan, design, code, test, package, and document specialized applications.

Two publications, *Developing an Application: Getting Started* and *Developing an Application: Related Documentation*, act as focal points to these publications. The following figure illustrates how these publications direct you to the VM/IS library and some key publications that you may need throughout the development of your application.



Developing an Application: Getting Started helps you design and integrate software packages for VM/Integrated System (VM/IS). This book introduces the VM/IS, its products, and its concepts and provides guidelines for packaging an application program. The book discusses tasks associated with developing an application package within the framework of VM/IS and describes special considerations for developing an application to work with a specific package or product.

Developing an Application: Related Documentation helps you quickly find the information you need and explain how to use the application development information library.

Use this book, *Developing an Application: Getting Started*, to define the tasks; then use *Developing an Application: Related Documentation* to locate the information you need to perform the tasks.

Who Uses this Book

This book is for anyone developing an application package for use in the VM/IS environment. It assists you in assessing tasks associated with developing an application.

This book is an introduction to the task of creating an application package, and is designed to be used with *Developing an Application: Related Documentation.*

How this Book is Organized

This book is divided into three parts:

- "Part I. Introduction to Application Development"

  This part presents an introduction to VM/IS and its concepts. It discusses the general task of creating an application and outlines the guidelines for packaging the application. In addition, a scenario illustrates the task of developing an application.

- "Part II. Stages in Developing a VM/Integrated System Application"

  This part discusses tasks involved in developing an application. Each major task is discussed in an introductory section, and then is broken down into subtasks. The task section ends with a task summary.

- "Part III. VM/Integrated System Packages, Functions, and Products"

  This part presents information on the grouping and release level of products within the packages that make up VM/IS and briefly describes the function of each product.

- "Glossary of Terms and Abbreviations"

  The glossary contains a list of terms and abbreviations that appear in the book. Any words that are italicized the first time they appear (like this: ***sample***) are listed in the glossary.

How to Use This Book

This book defines tasks involved in creating an application and points out considerations for each task. It presents a task-oriented view of the process of developing an application; in particular, it helps the person who may never have created an application package previously. Even programmers experienced in creating applications can obtain information about developing applications for VM/IS from the contents of this book.

After you use this book to define the tasks, use *Developing an Application: Related Documentation* to locate the information you need to perform the tasks.

The VM/IS BASE is composed of several functions that are functionally equivalent to other IBM products. This book uses the term **product** when talking about either BASE functions or optional package products.

Please note that the information provided here about these products is for guidance only. Specific product information is subject to change, and is only completely reflected in each product's library of manuals. For the definitive descriptions of a product's interfaces, datasets, macros, and other product information available for your use, always refer to the manuals provided with that product.

Contents

Part I. Introduction to Application Development 1

Chapter 1. The VM/Integrated System: An Application Platform 3
The VM/Integrated System 3
Installing and Migrating the VM/Integrated System 4
Tailoring the VM/Integrated System Menus 4
Learning about the VM/Integrated System 5
Using the VM/Integrated System 6
Administering the VM/Integrated System 8
Creating Applications for the VM/Integrated System 8
Adding Your Application Package 9

Chapter 2. An Application Development Scenario 11
Application Developed Locally: A Scenario 11

Part II. Stages in Developing a VM/Integrated System Application 19

Chapter 3. Planning for VM/Integrated System Application Development 21
Introduction to Planning 21
The System Architecture 22
Estimating System Resources 23
Required File and Data Structures 24
Transferring Data Between Products 24
System Prerequisites and Corequisites 24
Languages, Tools, and Utilities 25
Input and Output Routines, Facilities, and Techniques 26
Tailorable Interfaces 26
Methods of Communication 27
System Security Features 28
Data Recovery 28
Managing Your Project 29
Summary 30

Chapter 4. Designing a VM/Integrated System Application 33
Introduction to Designing 33
The Data and File Structures 34
The User Interface 34
Panel Layout 36
Display Station Program Function Keys 37
Panel Flow 37
Online Help and Tutorial Information 38
Commands 40
The Message Interface 40
Tailoring the VM/Integrated System – Productivity Facility 41
New or Tailored VM/Integrated System MAIN Menus 41
A VM/Integrated System Online Introduction Topic 41
VM/Integrated System Help Panels 42
The Application Program 42
Customizing an Existing VM/Integrated System Product 42

Tailoring the System for an Application 43
Disk Layout 43
DMKRIO, DMKSNT, DMKSYS, and DMKFCB Assembler Files 44
The System Directory 45
Other Files 46
Internal Communications in Your Application 47
Inter-System Communication Applications 47
Summary 47

Chapter 5. Coding a VM/Integrated System Application 49
Introduction to Coding 49
The Application Program 49
The Application Development Languages 49
Predefined Routine and Utility Languages 51
Using Two or More Programming Languages 52
The System Languages and Interfaces 52
The Development Tools of VM/Integrated System 52
Program Development Tools 52
Project Management and Control Tools 53
Text Editors 53
The User Interface 54
Customizing the VM/Integrated System – Productivity Facility 54
Tailoring Existing VM/Integrated System Products 55
Summary 55

Chapter 6. Testing Your VM/Integrated System Application 57
Introduction to Testing 57
Testing Your Application 57
Executing Your Application 58
Testing for Performance 59
Using Debugging Aids and Tools 59
Creating Test Data 59
Testing the System 60
Summary 61

Chapter 7. Packaging a VM/Integrated System Application 63
A Discussion of Packaging Tasks 63
The Package Tape Format 64
The VM/SP Installation Process 69
The Application Identifier File 70
The Installation Routine 70
The Verification Routine 74
The Memo to Users 75
Building the Package Tape 75
System Restrictions 77
Testing for Successful Installation 78
Summary 78

Chapter 8. Documenting a VM/Integrated System Application 81
Introduction to Documenting 81
The Publications Library 81
Designing the Publications to VM/Integrated System Standards 82
Writing the Publications 83
End-user Documentation 83
Administration Documentation 84
The Memo to Users 87

The Documentation Tools of VM/Integrated System 92
Summary 94

Part III. VM/Integrated System Packages, Functions, and Products 97

Chapter 9. The VM/Integrated System Products and What They Do 99
The VM/Integrated System Packages 99
The Base 99
Optional Packages 101

Chapter 10. What the VM/Integrated System Products Do 107

Summary of Changes 115

Glossary of Terms and Abbreviations 117

Index 125

Part I. Introduction to Application Development

Chapter 1. The VM/Integrated System: An Application Platform

In today's business environment, many organizations use computer systems to perform specific tasks. Controlling inventory, performing lab tests, manufacturing products, and administering the office are just a few of the many tasks for which computers are used. Each day, independent software development groups create a growing number of application solutions to satisfy this market area.

The IBM VM/Integrated System (***VM/IS***) is a powerful and flexible VM (***virtual machine***) system designed to help you create such specific applications. VM/IS offers a variety of program development languages, tools, and utilities that you can use to develop applications for VM/IS.

The VM/Integrated System

VM/IS is a prepackaged system that contains a basic set of IBM software products and applications. The system is open-ended and serves as an application platform that a customer can tailor to suit specific needs.

VM/IS groups its products into sets called ***packages***. Each package contains related products that can assist its users in completing their daily tasks. Figure 1 shows the package architecture of VM/IS.

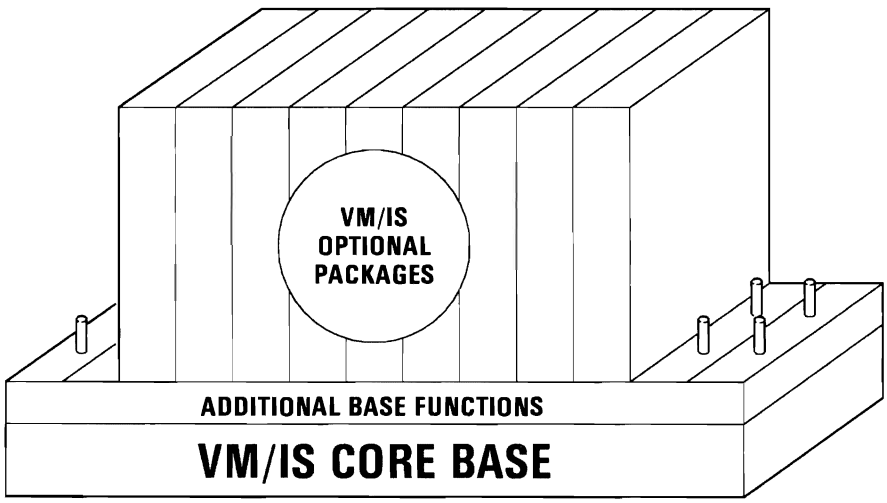


Figure 1. The VM/IS Package Architecture

VM/IS consists of the **VM/IS BASE package** and **optional packages** that can be installed on top of the VM/IS BASE. Each optional package contains a more specialized product set geared to a specific area of use. Packages are modular; therefore, VM/IS customers can order only those packages they require.

A series of ***menus*** and ***panels*** contains the interface to the BASE functions and the installed optional packages. In addition, VM/IS includes an online introduction to the tasks the customer can perform.

As a prepackaged system, VM/IS makes it easy to:

- Install and Migrate
- Tailor
- Learn
- Use
- Administer.

Such a system is attractive to potential customers because it reduces the amount of effort required to learn, use, and maintain the system.

Installing and Migrating the VM/Integrated System

VM/IS is designed to be installed by a customer with little or no system programming skills. VM/IS reduces the complexity of the installation process through the use of:

- A preconfigured system

  In the past, before installing a conventional VM system, the customer was required to design and implement a configuration that defined all hardware components in the system. This task, requiring trained personnel, could take a long time to complete.

  VM/IS is a preconfigured system. It supplies a number of predefined configurations. During the installation process, the customer chooses one of these configurations. Preconfiguration frees the customer from the complex task of designing a configuration and significantly reduces the skill level and time required to install the system.

- Step-by-step documentation

  VM/IS supplies documentation that "walks" the customer through the installation process. It gives step-by-step instructions that tell the customer what to do and how the system should respond.

- Customized software.

  VM/IS's customized software is packaged with its own installation procedure. This procedure reduces the number of steps and decisions a customer must make to install the system.

VM/IS customers can take advantage of software upgrades by ***migrating*** to new releases of VM/IS. Such migrations supply the customers with the most up-to-date release of VM/IS.

Tailoring the VM/Integrated System Menus

As already mentioned, VM/IS is an open-ended system. It contains an automated procedure for the customer to include your packaged applications on the VM/IS menus.

For example, suppose a company has bought an accounting package you developed. If you designed the package specifically for VM/IS, the company could install VM/IS and your package to create the system it needs. Figure 2 shows a simplified picture of the way the process works.

He now has enough information that he can design the application. He creates a flow chart of the program's overall structure, and writes ***pseudo-code*** for each of his application's modules.

Now John redesigns the VM/IS – Productivity Facility menus to incorporate the application. He identifies the changes he must make to particular files.

John feels he has finished the design process. Using VM/IS, he organizes everything he has done into a single document. He sets up a review meeting with Ann, who originally requested the application, and confirms the meeting by sending Ann a Professional Office System (***PROFS***) note.

The Review

The review meeting is attended by Ann, a few of her salespeople who will eventually use the system, John, and Janice, another programmer. John distributed copies of his design three days earlier to give everyone a chance to prepare for the meeting.

The salespeople ask for an online tutorial and a help panel for each application panel. They also ask what sort of documentation John will supply. Janice points out a few minor logic errors. Despite these problems, Ann says that she is very pleased with the progress John has made.

Most of the concerns that came up at the review are easy to address. John can fix the logic problems, design an online tutorial, and produce additional help panels without any problem. The documentation problem is tougher, though. He doesn't have the time or expertise to design a formal manual, and nobody else in his department does either.

Ann suggests that Brenda, who writes advertising brochures for Acme Appliances, might be able to help. John talks to Brenda and her manager, and both are agreeable. Brenda has never written anything larger than a brochure, and she thinks that writing a manual would be an interesting challenge.

John explains the project to Brenda, and shows her his design. They discuss the type of book that should accompany the application, remembering that they might have to cover two different areas: administration and end-user information.

After reading the documentation sections of *Developing an Application: Getting Started*, Brenda puts together an outline of her book.

While Brenda does this, John fixes his logic errors, and designs an online tutorial.

John shows his revised design and Brenda's outline to the people who attended the review. Everyone seems pleased. Ann wants to know when John will complete the project. He agrees to work out a schedule.

John returns to the *Developing an Application* series to see what it says about schedules. He discovers that a VM/IS product called Application System (AS) can help him with scheduling. Using AS, John and Brenda work out a project schedule.

Coding the Application

John is now ready to code his application. He codes the user interface first, so that he can have a working prototype on the system very quickly.

John begins a session in ***ISPF/PDF*** to code the interface. Using the edit facility, he codes the interface in the ***REXX*** system language. He uses REXX to match the structure of the VM/IS – Productivity Facility.

Within a week, he finishes the coding for the user interface. After adding a few dummy files, he has a working prototype. John does some preliminary testing of his prototype. VM/IS supplies several tools and guidelines to make testing easier.

Next, John begins coding the application. His design calls for a separate module of code for each function.

The customer information function, the inventory function, and the order function depend on each other. A salesperson can get information from one of these functions while he's in another function. This means that he can't do complete testing until all three functions are coded. Using ISPF/PDF however, he can test each individual section of code as soon as he finishes it.

The following day, Brenda approaches John with some questions. While writing the documentation, she has found that certain tasks would be easier if panels were rearranged and field descriptions were reworded. John promises to make the changes.

Several weeks later, John has finished coding the functions. He has a review meeting for the code, and a few minor errors are uncovered.

John asks Janice, another programmer, to design a test plan for his application. Because he has written the code, he shouldn't be the person to test it.

In the meantime, John includes the application in the VM/IS – Productivity Facility. He can test the changes without affecting any other users. After the application and its interaction with the VM/IS – Productivity Facility have been thoroughly tested, he makes the changes available to the users.

He now reviews Janice's test plan. It covers 150 possible test cases designed to test the limits of the application.

Brenda has now completed a draft of her book. She gives copies to several people and holds a review meeting. Everyone is pleased with the book, but it needs more detail. Several of the reviewers find small errors in the document, but no one finds anything major.

During the meeting, Brenda reveals that she has investigated the possibility of getting the manual printed. With her contacts in the printing business, and with the document prepared using VM/IS, she can get five hundred copies printed for a very reasonable price. In addition, the printed copies will match the rest of the VM/IS library. Everyone is pleased at Brenda's initiative.

Janice continues to test the application, and also tests Brenda's book. She finds a few errors, which are promptly fixed. Brenda sends her book to the printer.

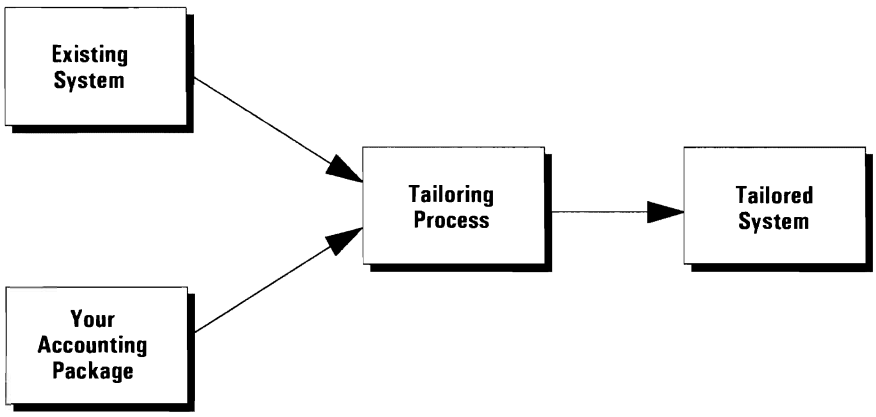


Figure 2. Building an Accounting Package onto VM/IS

The accounting package is added to VM/IS to form a unique system tailored to that firm's use.

VM/IS's design allows you to add your application through the use of ***panel tailoring files*** that you create. VM/IS uses these files to rebuild the ***dialogs*** and menus. The new dialogs and menus include your application as well as online help information and introductory dialogs that you create as part of your application.

Learning about the VM/Integrated System

VM/IS contains an online introduction dialog. The introduction dialog:

- Reduces the time required for a new customer to understand the system structure
- Reduces the complexity of selecting a desired application
- Quickly familiarizes the customer with the system.

The online introduction describes the tasks the customer can perform using the available applications on the system, such as creating presentation graphics, writing reports, and using the computer to communicate with others.

The online introduction is a hierarchy of topic and subtopic selection menus. Figure 3 shows the online introduction hierarchy.

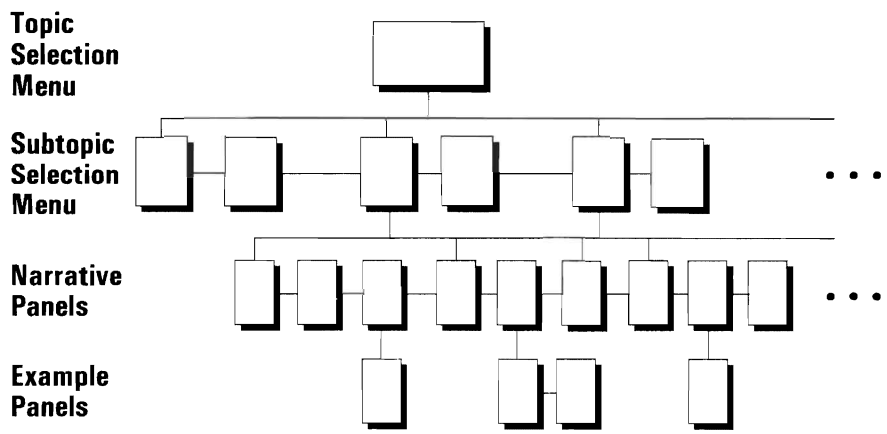


Figure 3. The Online Introduction Hierarchy

Each topic is a specific task, such as document creation or computer communication. Each topic may have a menu of subtopics. Each subtopic describes and shows examples of an application that lets the customer do the specific task discussed in the subtopic.

At the end of each topic, the online introduction invites the customer to try the described applications, without having to leave the online introduction.

The panels of the online introduction are arranged like the pages of a book. The customer can move through the panels — like turning the pages of a book. The customer can also select a topic and subtopic directly from menus — like using a table of contents and skipping directly to the desired page. Similarly, the example panels can be thought of as the illustrations of the book. The customer looks only at the illustrations that are of interest.

VM/IS also contains documentation that introduces the customer to the products of VM/IS. The documentation includes step-by-step examples that guide the customer through the first-time use of specific applications.

The online introduction and the product documentation help customers with little or no computer usage skills to become familiar with VM/IS quickly.

Using the VM/Integrated System

VM/IS makes its products available to the customer through the use of dialogs. Dialogs are series of panels that prompt the customer for information. The panels use the information to lead the customer to a particular application required to perform a specific task.

The dialogs reflect tasks that the customer can perform using the products and applications supplied by VM/IS. Menus are panels that let the customer select specific products to complete a task. The VM/IS MAIN MENU is shown in Figure 4.

```
                              VM/IS - MAIN MENU
------------------------------------------------------------------------------
To choose one of the following, type its number and press ENTER:

     1 Tools           Choose from a list of tools and applications.
     2 Personal        Your own menu that you can customize.
     3 Office          Perform administration and communication tasks.
     4 Document        Work with documents.
     5 Printers        Print files.
     6 Program         Computer programming.
     7 Database        Store and retrieve data.
     8 Graphics        Produce graphs.
     9 VMTasks         Work with CMS files and system resources.
    10 List            Choose from an alphabetical list of VM/IS functions.
    11 Introduction    Read about VM/IS functions.
    12 GO              Make option selections using a shortcut.


                        (C) COPYRIGHT IBM CORP 1988
------------------------------------------------------------------------------
PF: 1  Help     2  Short    3  End       4  Return    5  CMS       6  Go
PF: 7  ...      8  ...      9  Probrep  10  ...      11  ...      12  Retrieve

====>
```

Figure 4. The VM/IS MAIN MENU

Note: The MAIN MENU shown above may differ from the customer's MAIN MENU. The customer may have tailored the system by deleting products or functions or by adding optional packages.

Each menu presents the customer with a number of selectable items (such as Program for computer programming as shown in Figure 4). Each item refers to a particular task. The customer selects the appropriate task items to reach a particular application.

Figure 5 shows the structure of the menu hierarchy.

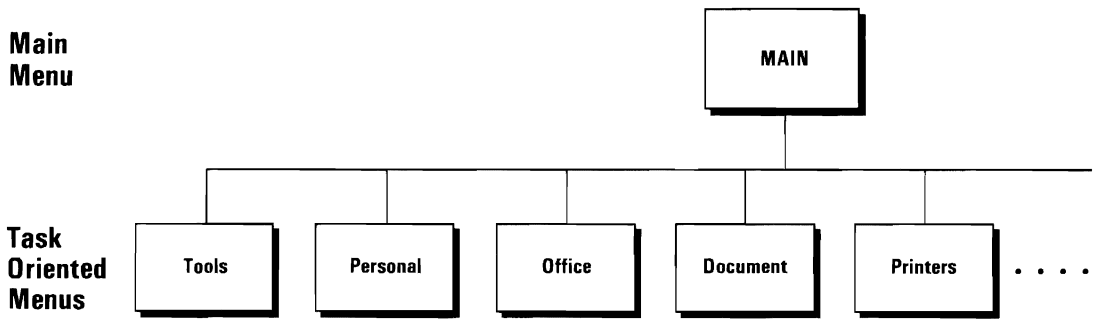


Figure 5. The MAIN MENU Hierarchy

When the customer selects a task from the MAIN MENU, a ***task-oriented*** secondary menu is displayed. Task-oriented menus list sub-tasks of the task the customer chosen on the MAIN MENU. The selectable items on these task menus lead to specific applications.

- Produce daily, weekly, monthly, or yearly sales reports.
- Submit an expense statement.

John agrees that the functions Ann wants are feasible. He now has enough information to begin his planning.

Planning the Application

John has never written an application before. He looks for useful books in the computer room's reference library.

He finds a pair of books from the *Developing an Application* series that look useful. John decides to investigate these books first.

He browses through *Developing an Application: Getting Started* (this book) and discovers that it was written for someone just like him. It offers the type of guidance he needs. The book even includes a scenario illustrating the development of an application. John decides to use this book to help him plan his application.

He picks up *Developing an Application: Related Documentation* and looks through the planning chapter. This book will help him find the additional information he needs.

Returning to his desk, John begins researching, using the *Developing an Application: Related Documentation* as his guide. Occasionally he makes notes.

He compiles a list of items:

- He must create a service virtual machine to handle application files. This virtual machine is logged on automatically (autologged) each morning.
- The default virtual storage (3M or 3072 Kilobytes of virtual storage) is enough for most tasks. Creating a sales report might require more storage, but only the manager would perform this task.
- He decides to use the Structured Query Language/Data System (***SQL/DS***) to store the data. The application must transfer data to and from SQL/DS, and meet all requirements of SQL/DS.
- John can think of no prerequisites other than SQL/DS.
- John decides to use ***VS COBOL II*** to code the application. VS COBOL II is well suited to do the job, and John is very familiar with it.
- In addition, John decides to use the Interactive System Productivity Facility (***ISPF***) as his dialog manager. VM/IS ensures that ISPF is running before the new application starts up.
- Some of the customer information is sensitive and must be protected. John decides to require the administrator to authorize each user to the critical data.
- Finally, John considers data recovery features. He decides to have data recovery at the ***transaction*** level. A confirmation is sent to the salesperson upon completion of each transaction. He also decides to use a ***mirror-disk*** technique, whereby the data is stored on two identical minidisks, each located on different disk packs. The system would update the first minidisk completely before updating the second.

John has now finished planning. The result of his research is a notebook full of notes. He organizes these notes, and enters them into a file on VM/IS.

Designing the Application

Now John begins to design the application. Once again, he looks to the *Developing an Application* series to point him toward the technical references he needs.

He makes a list of his design tasks:

- Design the user interface
- Design the file and data structures
- Establish the communication strategy
- Tailor the system by creating a service machine
- Design the application itself
- Redesign the VM/IS—Productivity Facility menus to incorporate the application.

He starts with the user interface, the most visible part of the application. He develops the screen flow first and then develops the screens themselves. He creates a hierarchical chart to illustrate the screen flow. (See Figure 8 for John's chart.)

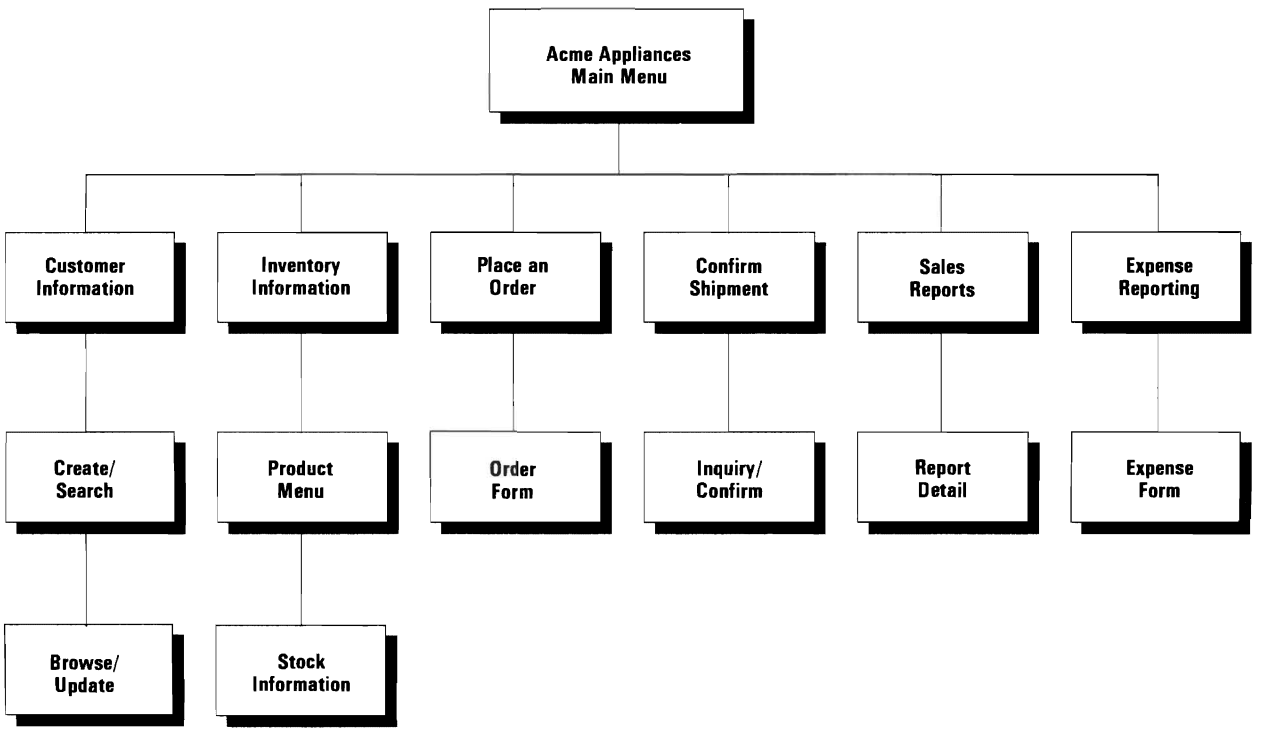


Figure 8. John's Hierarchical Chart.

Now John begins to design the actual panels for his application. He returns to the *Developing an Application* series for ideas on what to do next. Because he chose ISPF as his dialog manager, *Developing an Application: Related Documentation* refers him to the *ISPF Dialog Management Guide*.

He wants his panels to resemble panels within VM/IS, so he logs onto VM/IS to familiarize himself with the screen layout.

He designs the main menu, then the second level panels, then the lowest level of panels. (See Figure 9 for John's main menu.)

```
                          ACME APPLIANCE SALES
------------------------------------------------------------------------------
To choose one of the following, type its number and press ENTER:

     1 Information     Create/update/search for customer information
     2 Inventory       Check the stock on hand
     3 Order           Enter a customer's order
     4 Query           Query the status of a customer's order
     5 Report          Create a sales report
     6 Expenses        File your weekly expense report









------------------------------------------------------------------------------
PF: 1  Help     2  Short    3  End       4  Return    5  CMS       6  Go
PF: 7  ...      8  ...      9  Probrep  10  ...      11  ...      12  Retrieve

====>
```

Figure 9. John's Application's Main Menu.

He then designs the help panels. He decides that some of his panels need no explanation. He designs a help panel for each of the remaining panels.

John next designs the file and data structures. Because there is already an SQL/DS data base containing customer data, he doesn't need to design file and data structures for this part of the application. He must design data files for customer orders and expense statements. Both of these are SQL/DS data files. Using *Developing an Application: Related Documentation*, he finds out which SQL/DS books contain the information he needs. With this information, he can find the books he needs to design the file structure.

Because he plans to use a service virtual machine for the application, John investigates and designs the strategy for communication between users and the service machine, and between the service machine and SQL/DS.

He has already decided to use the ***VM/SP*** Inter-User Communications Vehicle (***IUCV***) as the communications strategy. Now he uses the *Developing an Application: Related Documentation* to locate the information he needs to design his strategy.

Next, he designs the service virtual machine to run his application. He makes a list of the minidisk links to include in the service machine's directory entry in the USER DIRECT file.

When Janice completes her testing, John installs the application on VM/IS. He runs PFSETUP to change the VM/IS menus, sets up the service virtual machine, and sets up user IDs for all the salespeople in Ann's department.

In addition, to distribute the application to other sites, John follows the directions in the *Developing an Application: Getting Started* to package the application to tape. He realizes that he has not written a Memo to Users to include on the tape. Together, he and Brenda write the document.

He and Janice do a final test to make sure that the application can be installed correctly. There are no problems.

The next day, the application is up and running in Ann's department. A week later, it is installed in another department. John receives congratulations from many people, including his manager, for a job well done.

Part II. Stages in Developing a VM/Integrated System Application

Chapter 3. Planning for VM/Integrated System Application Development

Before you can effectively develop an application, you must first have a plan of action. At the planning stage, you set objectives that describe what you want to achieve, and evaluate your current and future resources to ensure that you have the expertise and tools required to develop the application. From your observations, you then draw up a plan of action that describes what you are going to do.

Introduction to Planning

Planning is the first step in developing an application; you describe the application and the tasks it performs.

Planning requires knowledge and understanding of a large number of topics. You must have an understanding of the system for which you are producing the application and an understanding of the facilities available to assist you in producing it. You must understand how to structure the application code as well as the data used or produced by the application. If your application uses data from other products or passes data to other products, you must understand how to interface with these products. You must understand all aspects of communication, not only with other systems, but with other products in the same system, with the operating system, and with parts of your own application.

In planning your application, you probably start with the data processing concept: what the application will do with the data you supply or specify (the normal operation of your application; what happens to the data if nothing goes wrong when the application is running).

Next, think of your application's data storage capabilities and data or file structures. Will it use or store data in a ***flat file*** format or will it use a different format? How will the application access data (either its own data or data supplied by another application)? Can the application access the data in a multi-user environment (that is, can more than one user access the data at the same time)?

Finally you would consider the application's communication abilities. You must plan on having your application communicate with VM/IS (the system). The application may have to communicate with other applications (for example, SQL/DS), and it may have to communicate with other applications that reside on other systems. Depending on the way you design the application, it may have to communicate with itself; if you set your application up to run using multiple service machines, each part of the application may have to talk to every other part. Finally, you must determine how the application should communicate with the end-user; will you use a panel-driven interface (recommended) or a command-driven interface.

You must decide how the end-user will invoke your application. In addition, you must decide whether you will be using a ***saved segment***. Will you use ***transactional processing***? Perhaps you want your application to run only in ***batch processing*** mode. These considerations affect how your application programmers write the code.

Another aspect of planning concerns documentation. It is recommended that you plan your documentation concurrently with the other parts of your application.

Producing good documentation is a larger job than most people think. Consider having a staff whose sole task is planning, designing and writing the documentation to accompany your application. For more information on the task of documenting, see Chapter 8, "Documenting a VM/Integrated System Application" on page 81.

This chapter assists you in pointing out areas that require planning. The discussion comprises the following areas:

- The system architecture
- System resources
- Required file and data structures
- Transfer of data between products
- System ***prerequisites*** and ***corequisites***
- Languages, tools, and utilities
- Input and output routines, facilities, and techniques
- Tailorable interfaces
- Methods of communication
- System security features
- Data recovery
- Project management.

Remember that the biggest requirements in this phase is understanding what you can do and assessing the tools that are available for your use. You want to be able to make recommendations as to the direction in which you want your application development to proceed. To an extent, you are actually beginning to design your application; you are making decisions that determine how your application is built.

If, during the design part of your project, you discover that a planned course of action does not work, you can still change your plans to compensate without a great deal of trouble. During the coding or testing stages, changes become more difficult, more expensive, and more time consuming. Try to plan as thoroughly as possible for every eventuality. Good planning leads to good applications.

The System Architecture

System architecture is the specification of the relationships between the parts of a computer system. This definition refers to both hardware and software.

You may find it helpful to have an understanding of the system architecture with which you are working before planning your application.

You should understand the concept of a ***virtual machine*** and how it works to take advantage of all that it offers. You must understand the system components you can tailor to optimize your application's performance and what you can accomplish by modifying these components. You must understand how users can access applications or functions to take advantage of the best methods of access.

The operating environment of VM/IS is set up to share your computer resources. The system architecture is open-ended, allowing you to tune the operating environment to the needs of your applications.

VM/IPF is an administrative tool that you can use to help you modify or tailor one or more of these components of the operating environment of VM/IS.

Some of the things that you can tailor to meet your application specifications are listed below.

The system directory (the USER DIRECT file) defines all users (virtual machines) of VM/IS. The system directory defines the resources available to each virtual machine in terms of spool files, minidisks (storage), and other system devices.

You can tailor the characteristics of the operating environment by changing the contents of the system directory and certain DMKxxx "ASSEMBLE" files.

The ***DMKRIO*** file defines all Input/Output (I/O) peripheral devices of the operating environment. VM/IS contains a number of these ***configurations*** to choose from. Your application may require some special devices in the I/O configuration. You can define these by tailoring the DMKRIO file, adding new entries, or deleting unneeded entries.

The ***DMKSNT*** file permits the sharing of common program segments in one or more virtual machines. If your application uses saved segments, this file is tailorable, allowing you to make additions for your own application's saved segments.

The ***DMKSYS*** file contains a number of system macros that control many of VM/IS's functions or environment. This file is also tailorable, allowing you to change the macro options or definitions as necessary.

Estimating System Resources

You should understand the available system resources before you begin building your application. How much virtual storage do you intend to use for your application? Remember that the default minimum amount of virtual storage assigned each new user ID by VM/IS is 3M. If you plan your application to handle a large amount of data, you may want to ensure that the users redefine their virtual storage to meet your application's requirements.

In addition, plan to document a recovery strategy to cover the possibility that a user may try to run your application with insufficient storage. If the application goes into an error condition that requires the user to recover, the user may find that he or she has exited both from your application and from VM/IS. What does the user have to do to return to an area of VM/IS with which he is familiar? Remember that a large percentage of users would not be familiar with a ***CMS*** command-oriented environment.

You should consider the amount of storage space (direct access storage device, or ***DASD***) that your application might require, and where (on which DASD volume, and on which minidisk) you plan to store it. You may decide to define a new minidisk specifically for your application.

You may want to consider the amount of processor time your application requires to run. If the application requires a great deal of processing, your application may take up such a high percentage of processor time that other applications are affected. You may want to adjust the function of the application to compensate for this problem or even consider using a batch approach.

Required File and Data Structures

If you are designing your application to be independent of other VM/IS products, you probably want to design file and data structures to fit the VM/SP environment.

On the other hand, if you decide to use the resources of other VM/IS products, either to store and retrieve data or to process data, you must understand the file and data structures used or required by those products.

In addition, if you use the resources of more than one VM/IS product, you may need to understand several different file and data structures. You may also have to be able to modify the structure of the output from one product to meet the input requirements of another product.

Transferring Data Between Products

Previous sections mention the idea of using the resources of other VM/IS products. These products can be installed on the same system or on another VM/IS system with which your system communicates. If you do decide to use other product resources, you must understand how to transfer data between products, and you may have to understand how to transfer data between systems.

Many of the products of VM/IS contain specific paths of data transfer or an interface to one or more of the products of VM/IS. You can only use the resources of another product if you understand how to transfer data between your application and the other product. Remember that many VM/IS products use service machines. To transfer data to and from these products, you must also understand aspects of communication. For more information about communication with other products or systems, see "Methods of Communication" on page 27.

System Prerequisites and Corequisites

In planning for your application, you may decide to use the resources of another application (for example, SQL/DS) to perform a function. This method gives you several advantages:

- You save time and money because you do not have to produce the code for that function.
- Because you do not have to code the function, you do not have to test the function itself. Your testing need only include any code you must write to interface with the other product.
- Depending on the function or product you use, you may be able to save the task of designing file and data structures, using the same format as that used by the other product.

With the advantages, however, there are disadvantages and potential problems:

- Although you do not have to test the function of the product you decide to use (the owner of the product will have tested it before he made it available to be used), you must ensure that the function gives you the results you need for your application. If you do not get the expected results, then you may want either to code the function yourself to produce the right results, or to use a different product or combination of products. For example, suppose you want to take

specific data from a data base, format the data into a chart, and print a copy on your system printer. SQL/DS does not perform all these functions by itself. You can design your application to request data from SQL/DS, use VS FORTRAN to format the data into a chart format, then use ***VM3812*** to print the data.

- You must produce the code to interface with the other product or products and ensure that the code produces the correct results.
- You must ensure that any required products are installed on your own system for purposes of designing, building, and testing your application. You must also ensure that your application programmers have sufficient understanding of the required products to be able to interface with them, and that you have the correct publications for reference when they are needed. You must also ensure that potential customers for your product are aware of any prerequisites or corequisites.

In addition to software prerequisites or corequisites, your application could require specific hardware. If your application uses graphics functions, you may need graphics terminals. You may need a specific type or size of printer. Once again, you must ensure that you have the appropriate hardware installed on your own system for purposes of designing, building, and testing, as well as to ensure that potential customers are aware of the need.

Languages, Tools, and Utilities

Depending on the packages you have installed, VM/IS gives you access to several programming languages, tools, and utilities to help you build your application. You have at your disposal several program development facilities and a wide choice of programming languages, including:

- ***Assembler***, contained in the VM/IS BASE package
- ***VS COBOL II***, contained in the TXN package
- ***REXX*** (Restructured Extended Executor language), contained in the VM/IS BASE package
- ***VS FORTRAN***, contained in the Engineering/Scientific Program Development Support (E/SPDS) package
- Other languages that run under VM/SP and included in the VM/SP system offering, including:
  - ***APL2***
  - ***IBM BASIC/VM***
  - ***Pascal/VS***
  - OS PL/I V2.

For information about the programming languages available with VM/IS, see "The Application Development Languages" on page 49. VM/IS also includes a large number of predefined subroutines to help you develop your application.

By planning the development tools, the available subroutines, and the available languages, you can optimize your project. Try to plan your application to use a programming language with which your programmers have expertise. In addition, try to plan your application around existing subroutines to save coding time.

Input and Output Routines, Facilities, and Techniques

You should understand the routines and facilities that are available to control your application's input and output.

For example, you can use REXX, EXEC2, and XEDIT macros that are supplied with VM/IS. You can also make use of normal CMS commands, as well as macros, to control a reader, punch, or printer.

You could use ***CP*** facilities including the Inter-User Communication Vehicle (***IUCV***) to communicate with other virtual machines, with a CP system service, and with yourself.

Another facility is the message system service. This facility allows an application to trap the console output of a virtual machine.

You could decide to use the single console image facility. It allows you to specify that all console input or output for a designated user ID (for example, a disconnected service machine) be passed to another user ID. Using this facility, a logged on user ID can remotely control a disconnected service machine.

The logical device support facility allows an application to define a logical device, which CP treats as a real device. The logical device support facility traps any input and output involving that logical device for use by the application. An example of an application that uses this facility is the VM/Pass-Through Facility (***PVM***).

It is recommended that you use the CONSOLE macro to control the output to a customer's terminal screen. Using this macro allows you to use fullscreen CMS for your application.

Another facility that can be used is the DIAGNOSE instruction. This instruction gives you a variety of CP facilities with which you can work. For example, you can use the instruction to get the date and time, to check the type and model of a device (perhaps a terminal), or to check the amount of virtual storage in a virtual machine.

Some other facilities you can use include the following macros:

CMSIUCV CMS IUCV support
DMSFREE CMS free storage management
DMSKEY CMS storage protect key management
HNDIUCV CMS IUCV support
LINERD Reads a line of information from a terminal. This macro is similar to the RDTERM macro, but provides increased flexibility. This macro is used for "line mode" input and is compatible with fullscreen CMS.
LINEWRT Writes a line of information to a terminal. This macro is similar to the WRTERM macro, but provides increased flexibility. This macro is used for "line mode" output and is compatible with fullscreen CMS.

Tailorable Interfaces

In planning your application, remember that some of the products packaged under VM/IS can be customized. This means that you can enhance or modify the basic operations of these products to enhance the operation of your own application.

For example, you can tailor the VM/IS user interface (the VM/IS – Productivity Facility) to rearrange the order in which your application appears on the panels. You can modify the System Profile ***EXEC*** (SYSPROF EXEC) to have each user go directly into your application. You can tailor the system so that your application is accessible from the MAIN MENU, or you can arrange to have your application called by issuing a command.

Methods of Communication

Your application needs to communicate in several different ways. For example, the most obvious is communication with the user. This means designing an interface. You can use either a panel-driven interface or a command-driven interface.

It is recommended that you use a panel-driven interface because VM/IS users may not be familiar with working in a command-driven environment. If you decide to use a panel-driven interface, you must plan your screen format and content. Often one panel can replace several commands. You can use your hierarchical designations of commands, to establish your panels. For example, you could group all primary commands on a single panel. Your secondary panel could replace commands of a secondary importance or that perform a secondary function, and so on.

Whichever method you decide to use, you must still establish a set of commands to control your application. You should also decide on a hierarchy of commands, dividing the commands into a list of primary commands, secondary commands, tertiary commands, and so forth. For example, a primary command would either initiate your application or initiate one of the functions of your application. A secondary command would perform a task within the application or function, but would not work until one of the primary commands had been issued.

The user interface is only the first communication area you must address. In addition, you must consider communicating with the operating system (VM/SP). To exist in the VM/IS environment, your application must be able to operate under VM/SP. Among the considerations here is the question of using a saved segment. Consider using a saved segment if your application is used concurrently and continuously by several users.

In addition, your application may have to communicate with other applications. You will know whether you must be communicating with an application's service machine, or whether there are other protocols you must follow.

You may even have the situation where you must communicate with another system. In this case, you have to establish whether the system is a VM system, or whether it uses a different operating system. You have to determine the application with which you are to be communicating on the other system, and how you can receive the output from that application.

VM/SP contains the Transparent Services Access Facility (***TSAF***) to allow you to communicate easily with another VM system also using TSAF. TSAF uses the Advanced Program-to-Program Communication/VM (APPC/VM) facility, a facility similar to IUCV. The facility is not apparent to the end-user and is simple to administer. You can use TSAF to access resources, such as data stored in a data base that is resident on another system, or even a physical resource like a tape drive that is attached to another system.

Other facilities you can use in communicating with another system include the Remote Spooling Communication Subsystem (***RSCS***) and the Systems Network Architecture (***SNA***), which is implemented through the Virtual Telecommunications Access Method (***VTAM***).

System Security Features

Security is an important feature for which to plan. You should not design an application in which any user can modify or delete important data. Carefully determine the type of user who should have access to the data. VM/IS is packaged with several security features. The simple act of logging on to the system demonstrates the most important security feature: each user must have his or her own personal password to gain access to the system. In addition, each user must change this password on a regular basis to retain access.

If your application uses one or more service virtual machines, you may decide to ship sample directory entries for these service machines. It is recommended that you ship any predefined user IDs with the passwords set to NOLOG to ensure the security of your service machines. Remember to include information on changing the passwords in your installation instructions.

If your application is intended for processing sensitive data (for example, confidential salary data), you would want only certain types of people to have access. In addition, you might want the type of access to be tailorable by the customer.

You have several ways of limiting access to the application. For example, you could simply design the application so that a user requires a special password to gain access. However, this method could become cumbersome, and any user who discovered the password could then gain access. It might be easier and better to assign a security level to the command that accesses the application. This way, only those users who have the correct security level (or higher) can gain access to the application.

But accessing the application is only the beginning. You also need a method of securing your data, especially if you are using functions from other products, each of which may use its own security features. When using other products, ensure that the security offered by that product is sufficient for the needs of your application. You must check that the product you intend to use meets the requirements of your application.

Data Recovery

Plan your application around the possibility that, at some time when a customer is using it, the system on which it is installed may shut down abnormally. This shutdown could be caused by something as simple as a power failure. Your application should have the facility to recover as much data as possible from the interrupted session.

In addition, occasionally users interrupt a session by accidentally pressing the wrong key or combination of keys. This action often occurs after the user has entered a large amount of data. If your application has no means of recovery, affected customers could lose a great deal of data.

A recovery feature could be as simple as the autosave feature implemented by the System Product Editor (XEDIT). The editor automatically creates a secondary file containing changed or additional information. The user can then decide how often this backup file is updated. If the system goes down, or an editing session is interrupted, the user does not lose all his or her data.

Once again, other products you use could well have their own recovery features built in as part of their normal process. It is your responsibility to investigate these recovery features to determine if they meet your needs. If they do not meet your requirements, you might want to consider augmenting the existing recovery features, or using a different product.

Another area to consider is backing up your application and the data used by your application on a regular basis. What would happen to the application and the data used by that application if a disk pack failed, leaving all application files inaccessible?

You can plan to have your application data backed up to tape or to another minidisk. If you decide to back up your application data to tape, consider setting up a suggested backup schedule for the customer. If you choose to back up to disk, choose a minidisk located on a different disk pack and consider an automatic backup procedure.

Another product that runs in the VM/IS environment is VMBACKUP Management System. VMBACKUP is a combination of a minidisk dump and restore system and a CMS file archive and retrieval system. The dump and restore system is used to make backup copies of CMS and non-CMS minidisks periodically. The data is copied to physical tapes or to DASD where it is available for retrieval if needed. The archive and retrieval system allows the user to copy CMS files to a storage area and to retain those files for a user-specified time.

VM/IS includes a service machine called SYSDUMP1 designed specifically to handle system backups. You can use this virtual machine to handle your application's backups, or you can design another strategy. You should recommend a backup schedule to the customers, especially if your application's data changes on a regular basis. Specify the files the customer should consider backing up (or the types of files if this information can be modified by the customers) and how often the customer should consider backing them up. You might include as part of your application a facility to automatically back up application files to a mirror-disk on a regular basis.

Managing Your Project

You could have several programmers working on your application as well as planners, testers, and writers. You must constantly coordinate to ensure that everyone works in an organized manner to meet the objectives by the scheduled date of completion.

By the time you finish planning, you should have estimates of the time required to develop each section of the application. You may even expand your estimates into actual dates by which time you expect a specific part of your project to be completed. You should establish a schedule based on these estimates and update it as events progress.

Many of the events must run concurrently. For example, if you have several programmers, you will want each to work on his own section of code. In fact, depending on the function you plan for your application, you may decide to assign several programmers to each function.

You can use the Application System (AS) to assist you in establishing, and maintaining your schedule.

VM/IS supplies you with other tools to assist you in controlling your project in other ways. For example, use Professional Office System (***PROFS***) notes for your communications. PROFS allows you to organize notes that you send or receive into note logs for easy retrieval later. In addition, PROFS offers the calendar facility to help you schedule your time for meetings, appointments, and reminders. You can schedule meetings for one or more days up to a year in advance. You can also schedule repetitive meetings (for example, a weekly project meeting).

You should familiarize yourself with the tools available and plan on utilizing them to assist you in controlling your project.

Summary

To plan a VM/IS application successfully, you should:

- Understand the system architecture and the tailorable files, including:
  - The USER DIRECT file
  - The DMKRIO ASSEMBLE file
  - The DMKSNT ASSEMBLE file
  - The DMKSYS ASSEMBLE file.
- Understand the system resources that are available for your application. You should consider the following:
  - The amount of virtual storage required to run your application
  - Recovery if a user runs the application with insufficient virtual storage
  - The amount of DASD storage required for your application
  - The amount of processor time required to run your application.
- Consider the design of the file and data structure of your applications. If you are using other applications, understand the file and data structures used by those applications.
- If you decide to use the resources of other applications, understand how to transfer data to and from those applications.
- If you decide to use the resources of other applications or require specific pieces of hardware, make sure that the applications or hardware satisfy your project's needs. Make sure that the customer is aware of the additional requirements to run your application.
- Be aware of the languages, tools, and utilities available with VM/IS to assist you in designing and building your application. Plan to use as much of the available resource as you can to make your job easier.
- Understand how to modify the VM/IS interface to include your application. Understand how to modify (if necessary) the tailorable parts of other applications to improve performance.

- Plan your methods of communication in the following areas:
  - The user interface
  - The operating system (VM/SP)
  - Other applications
  - Other systems.
- Be aware of the security offered by VM/IS and its component products and plan for additional security if your application handles data of a sensitive nature.
- Plan a backup and recovery strategy against the time when the system, the application, or the hardware fails, causing loss of data.
- Establish and update a project schedule. Try to estimate the time needed to complete individual sections or phases of your project. Set target dates for completing each section of the project and for completing the project itself.

Chapter 4. Designing a VM/Integrated System Application

After you have established your goals in the planning stage and created a schedule, you can then determine how you can create your application. Based on your knowledge of the system with which you are working, you can:

- Design your application program
- Design the data and file structures necessary to support your application
- Design a user interface for your application
- Decide how to tailor the operating environment to suit your application's needs
- Decide how your application fits in with the VM/IS panels
- Design the documentation.

The above tasks do not have to be carried out in the order in which they appear. The only requirement is that, in the end, you have designed your entire application. When you finish, you should have a set of specifications that your programmers can follow when doing the actual coding.

Introduction to Designing

In the planning stage, you investigated tools and procedures, data formats, system interaction, communication, and so on, finally deciding on the implementation of your objectives of your application. In the design stage, you take the plans and expand them into what eventually ends up as your application. Look on the design stage as a highly detailed planning stage where you design the specifications for your application.

Take the information gathered and evaluated during the planning stage and expand it into an actual design. For example, in the planning stage, you investigated data and file structure and decided on one or more structures to implement. In the design stage, you take those recommendations and begin to establish the fine details of the structure. You must determine that a course of action is feasible and design everything a programmer will need to produce the code that establishes that file or data structure. In the coding stage, all that will remain will be coding to the specifications you design in this stage.

If, during your designing stage, you decide that a particular structure recommended at the planning stage is not feasible, return to the planning stage to reevaluate file and data structures and make a new recommendation. Remember that it is much easier to change paths during the design stage than it is during the coding, testing, or ***packaging*** stages.

Another aspect of design concerns documentation. It is recommended that you design your documentation concurrently with the other parts of your application. Producing good documentation is a larger job than most people think. For more information on the task of documenting, see Chapter 8, "Documenting a VM/Integrated System Application" on page 81.

The Data and File Structures

During your planning phase, you investigated potential data and file structures and decided on specific structures for your application. During the design phase, you design an actual structure that fits the requirements of the planned structure. You work out the fine details and determine whether such a structure is feasible. If you cannot design your data and file structure to fit, you must then return to the planning stage to reevaluate the requirements of your application.

In addition, you probably want to use a component code to identify your application files. A component code is a three-character string that identifies your application's files and components. Named objects such as files, panels, messages, and modules are usually made up of the application component code followed by a unique identifier within your application.

For example, one of the VM/IS component codes is ESD. Some files that are part of VM/IS that use this component code are ESDPLIB MACLIB and ESDGRAPH ESAITEMA.

Each IBM application has a unique component code; each application installed on a VM/IS system, including your application, must have a component code that is unique when compared to other VM/IS applications.

Choose component codes that are unlikely to be duplicated by other application developers. Since IBM component codes do not usually contain numerals, consider always including at least one numeral in the component code you choose. You can also use the special characters at sign (@), pound sign (#), and dollar sign ($) in your component code to further decrease the odds of duplicating another application's component code.

After you have your component code, you must ensure that your named objects all start with the component code. If your named objects do not start with the component code, you run the risk of one of your panels or files having the same name as one of those supplied by IBM, another application developer, or your customer. That panel or file could appear when your panel or file should, or could be displaced by your panel.

If you need to store your data, depending on the needs of your application, you may want to investigate the features of SQL/DS.

The User Interface

Any VM/IS application that you develop should have several functional and design characteristics. Your application should satisfy the following:

Functional Characteristics

- Design your application so that it is interactive and panel-driven as opposed to line-driven (command-driven).

  Line-driven applications require that the customer enter commands one line at a time to perform a given task. Panel-driven applications allow the customer to enter data in response to prompts or to select items from **menus**.

  While line-driven applications operate properly with VM/IS, applications are more user-friendly if panel-driven.

- Invoke your application by an EXEC or module that does not require any user-supplied parameters.

  VM/IS usually starts an application by invoking an EXEC. If your EXEC requires user-supplied information, design the EXEC to display a panel to obtain the required information.

- Use a panel layout similar to that used by VM/IS.

  Where possible, set your program function (***PF***) key functions so that they are the same as the PF key functions on the VM/IS panels. For example, define PF1 to request your application's HELP function.

- Develop your application using the current release of VM/IS.

  You should be aware of enhancements to the VM/IS products that can affect your application. You can obtain this information in *VM/IS Planning for Your System.*

Design Characteristics

- You should use the Interactive System Productivity Facility (**ISPF**) as your **dialog** manager because of the improved productivity it offers. Also VM/IS will ensure ISPF is running before starting up your application from the VM/IS menus.

  Regardless of the dialog manager you use, your application must not affect the ISPF environment in which VM/IS operates.

- After VM/IS starts ISPF, no other application can start ISPF. If you designed your application using ISPF, and you want to start it with the same EXEC in either the CMS or VM/IS environment, the EXEC should then check whether the ISPF environment is already started.

- If you use ISPF to create dialogs for your application, you must define the files that contain your application's libraries.

  There are two methods you can use to define these files to the ISPF function:

  - LIBDEF service
  - FILEDEF command.

  Using the LIBDEF service of the ISPF function rather than the FILEDEF command tends to improve overall system performance. The LIBDEF service defines the files that contain your application's libraries while the ISPF function is running.

  For more information on the LIBDEF service, refer to the *ISPF Dialog Management Guide*, or to *ISPF Version 2 Dialog Management Services and Examples.*

If you decide to use the FILEDEF command, your application requires FILEDEFs for its MACLIBs before ISPF can be initialized. VM/IPF will perform these FILEDEFs for you using the DTRFDEF function. The FILEDEFs are performed if your application's libraries follow these naming conventions:

```
xxx$PLIB MACLIB  -  Panel library
xxx$MLIB MACLIB  -  Message library
xxx$SLIB MACLIB  -  Skeleton library
xxx$TLIB MACLIB  -  Table library
xxx$XLIB TXTLIB  -  Text library
xxx$LLIB LOADLIB -  Load library
```

where xxx is your application's component code, and $ is any valid CMS filename character or the null character.

An alternative to the naming conventions is that you can create a FILEDEF file.

More information on the DTRFDEF function is found in the *VM/IS Tailoring Your Menus*. You can find details on the FILEDEF command in the *ISPF Dialog Management Guide*.

- Avoid requiring the customer to perform extensive tailoring to your application. If tailoring is required, you should pretailor your application as much as possible to simplify the installation process.
- If your application requires additional minidisks, be careful to link and access virtual addresses and filemodes not used by other applications.

  You can use the GETFMADR module to help you choose unused filemodes and virtual addresses. For more information about the GETFMADR module, see the *VM/IPF System Reference*.

Panel Layout

Try to make your panels similar in appearance to the VM/IS panels. You should give each panel a title and identification number and list PF key definitions at the bottom of the panel. If your panel requires user interaction, try to supply a help panel to accompany it, invoked by a PF key.

Try to design your menu panels to look like VM/IS panels. We suggest a numbered list of items from which the customer can choose, with a command line at the bottom of the panel. You may want to group related items, separating them from other groups with a blank line. To proceed to the next panel, use the item number or a keyword. You may also decide to use a PF key to choose the function, as does PROFS.

Information panels should contain only related fields. Try to avoid using panels with so many fields that the panel looks cluttered or crowded. You may decide to supply default information in the fields which can be overwritten if necessary. If the type of information entered by the user most likely remains the same, you can set up your application to remember the information entered by a user. Your application can then display that information in two ways:

- On subsequent panels requiring the same information
- On the original panel the next time the user displays it.

This design could make your panels more user-friendly by reducing the amount of information that must be entered each time.

Remember to list active PF keys at the bottom of information panels and to supply a help panel for each one.

Display Station Program Function Keys

VM/IS consists of many products, each of which has its own PF key definitions. This fact creates a problem when you start defining your application's PF key settings. You have three choices:

- Design your PF key settings after the settings used by VM/IS.
- Design your PF key settings after the settings used by the applications with which your application interfaces.
- Use your own PF key settings in a format useful in the context of your application.

It is recommended that you design your PF key settings to be like the settings used in VM/IS, as they appear on the MAIN MENU and subsequent panels. For example, the MAIN MENU defines PF1 to access the help function. Remember to design your PF key settings both for consistency and to save confusion on the part of the customer.

If your application interfaces with other applications, you may want to make your PF key definitions similar to those definitions used by the products with which your application interfaces. Consider this strategy if your users will be using your application more than fifty percent of the time. For example, if your application interfaces with PROFS, you might want to define PF9 for your HELP function rather than PF1.

You should define your PF keys to perform the following minimal functions:

- The HELP function
- Return to the previous panel
- End the application.

You should define other PF keys to fit your application. Remember that you do not have to define all PF keys. You may decide to supply a tailorable file in which the customer can modify the PF key settings to suit his own installation. If you decide to supply a tailorable file, remember to document the fact that the file exists, and how to modify it.

Panel Flow

If you are using a panel-driven interface, the panel flow (that is, how the panels are organized) is important. Generally, you should plan to organize your panels from general topics to specific topics. In other words, your first panel could be a menu that contains a list of items to cover all major topics or functions in your application. The next level should be even more specific, and so on.

Plan your panel flow as a flow chart to ensure that the panels are organized logically.

Online Help and Tutorial Information

You should supply online help information as an aid to users of your application; you should also supply an online tutorial to assist new users to familiarize themselves with your application.

As a minimum, you should plan to supply online help information for each interface panel in your application. If your application contains other panels (for example, work panels that normally appear blank until the user enters information), you should supply help information for that area of the application.

You can use the windowing facility of VM/SP to make maximum use of your help and tutorial information. You can define either a small window to overlay part of the current panel or a large window to overlay the entire panel. In addition, you can make the contents of the tutorial window interactive to help the user learn how to use your application.

Generally, an application contains three types of panels:

- Menu
- Information
- Work.

The content of the online help you should supply is listed below for each type of panel.

Online Help for Menus

Generally, a menu is a panel consisting of a number of choices, from which the user chooses. For this type of panel, try to explain the choices. For example, the first panel in your application may be a menu that lists several functions available to the user. The function description as listed on the panel may not tell the user everything the function can do. The online help should address this situation by explaining each function that is listed.

In addition, the online help should explain how the user is to proceed to the next panel. If your panel utilizes a field into which the user must enter information, tell the user the type of information he or she can enter into the field, and the form in which to type the information. For example, you may have designed the panel to accept either a number or a keyword. Explain the requirements of the field (for example, the field may accept only up to eight characters, or may accept only numbers), and what must be done to proceed to the next panel (for example, press ENTER, or a PF key).

Regardless of the method you choose, be sure you explain the method of selection clearly and completely. In addition, you might want to explain what other information the user can enter on the command line. For example, you may have a method of combining commands to access a specific panel quickly (a fast-path, equivalent to the VM/IS ***Go Facility***), or perhaps the user can execute CMS commands from this panel.

Finally, list the active PF keys and explain their functions.

Online Help for Information Panels

Information panels can vary greatly in format and content. Generally, an information panel asks for information from a user that may be needed to advance to the next panel. This type of panel can consist of one or more fields of varying lengths and formats.

Many fields on an information panel are optional, and many fields use default information if not filled in. In addition, default information that can be changed can appear in some or all of the fields. You should explain what information is required for each field, what format this information should take, and which fields are optional. If the field defaults to specific information, state the default.

If the fields require input of a specific size or content, state the requirements. For example, if the second field looks for a maximum of sixteen alphabetic characters, explain this. If the field descriptor does not adequately explain what type of information is needed, provide expanded information to describe the field.

You should consider stating the overall purpose of the panel, if this is possible. Finally, as with other panels, list active PF keys and their functions.

Online Help for Work Panels

A work panel is a panel on which a user can accomplish a specific task. It is usually the final panel in a series of menus and information panels.

Work panels can vary in appearance all the way from a totally blank panel up to a panel that contains some information that must be expanded, changed, or deleted. The panel could list active PF keys and could have a title.

The chief aim of online help for a work panel is to indicate the intent of the panel. In other words, what can the user do using this panel? This explanation could become very complex, depending on the function and the type of application. Try to be as precise as possible, but remember that the manuals often explain (probably with examples) what can be accomplished from this panel. You do not have to try to duplicate the information in the manuals, but rather give the user the immediate information needed to accomplish his tasks. Remember that he or she can always refer to the manual for additional information.

Once again, remember to list active PF keys and their functions.

Providing Tutorial Information

If you decide to provide an online tutorial for your users, you should plan on adding the topic to the VM/IS Introduction Topics Menu series of panels. You should include an overview of your application and as many specific examples as you feel is necessary to illustrate the function of your application. If your application used predefined forms or formats a report to appear in a specific order, illustrate both the filled in input panel as well as the final result.

You may even want to design a tutorial session to teach new users about your application. You can design the tutorial to appear to be interactive. In other words, while the user enters responses on the tutorial panels (panels that simulate your application's panels), the panels do not actually change any application data. If you use an interactive format, you should include reminders on each panel to point out to the user that this is only a learning session, and does not actually interact with the application.

In addition, you may want to include the facility to go directly into the application from your tutorial, so that users can take advantage of their knowledge immediately.

Commands

It is recommended that you design your application to be panel-driven rather than command-driven. You may want to supply the facility to bypass your interface. If you decide to do so, remember that you must follow all standard rules in establishing VM/CMS commands. For example, your commands cannot exceed eight characters, nor can you use as a command any sequence of alphanumeric characters that already exists as a command in VM/IS or in VM/SP.

If you use a command interface as the only interface, ensure that all commands are adequately documented. If you design commands as a supplement to your panel-driven interface, you should still document the commands available, but recommend that the customer access your application through the panel interface.

Whatever type of interface you use (panel-driven or command-driven), you should take advantage of VM/SP's parsing facility. Using the Definition Language for Command Syntax (DLCS), define your command syntax in one or more CMS files to have VM/SP check commands for proper operands, options, keywords, and so on. Using the parsing facility eliminates the need for you to check command syntax as part of your application. For more information, see your VM/SP documentation.

The Message Interface

Your application must display messages at some point when it runs. Whether these are error messages, information messages, or warning messages, you must determine how to handle them and how to communicate their meaning to the users. If you are using a panel-driven interface, you may decide to define a specific area of the panel as a window to display messages. Further, you may also decide to highlight the messages to draw attention to them.

Both ISPF and VM/SP include a message handling facility you can use. If you are using ISPF as your dialog manager, you will also use ISPF to display messages; ISPF displays these messages directly on its panels. For non-ISPF messages, it is recommended that you use the CMS message facility to transmit messages to users.

However you decide to handle your messages, you must establish a numbering convention for them. Your numbering should include the following items:

- A prefix code to appear as the first part of the message number for all application messages. It can be from one to six alphabetic characters long, and must be unique to ensure that the correct message is displayed when your application calls a message. For example, you might consider including your component code as the first three characters of your prefix code.
- A message number composed of three numeric characters.
- Optionally, a suffix to indicate the type of message (information, warning, or error).

Remember that you can store all your messages in a single file, rather than code them into your application. This method makes it easier to access messages and, if you decide to translate your application into another language, makes translation easier.

In addition, you should document your messages, especially if the messages are visible to the users. In the documentation, you should include the text of the message, an explanation of the message, what VM/IS has done about the message, and what the user should do as a result of the message.

Tailoring the VM/Integrated System – Productivity Facility

For VM/IS to recognize your application and include it in the VM/IS panel interface, you must create panel tailoring files that cause VM/IS to change its panels to include your application.

VM/IS includes the facility to add, change, move, or delete items from the VM/IS – Productivity Facility panels. Using this supplied facility, you can modify the VM/IS panels to give your application the degree of prominence you desire.

The following sections discuss how to tailor the VM/IS user interface. You can find more detailed information in the *VM/IS Tailoring Your Menus* manual.

New or Tailored VM/Integrated System MAIN Menus

You will have decided during your planning stage from which level of panel you want your application accessed. You may want to insert a menu option into the MAIN MENU to directly access your application. If you do so, it is recommended that you follow the ***task-oriented*** format of VM/IS when adding your menu option.

Remember that the library option is already in place on the TOOLS menu. You may decide to use this option as the facility by which you make your application available to the users.

Before you actually make changes to the panels, you should plan how you want them to look. Understand the initial panel appearance, sequence, and structure. Decide how and where you want your application to appear. If your application is designed to address a specific task, you may want to add the application to an existing task area. For example, if your application performs office tasks, you may want to include an entry in the OFFICE MENU. On the other hand, you may decide to use the existing Library facility to access your application.

Whichever way you decide to proceed, plan ahead. You might want to design your modified panel structure on paper before trying it online. You must create new files and modify other files. Then VM/IS takes the information you supply to build new panels. While you can always repeat the process if you dislike the final result, the process can be time consuming if you must repeat it many times.

Remember that you may have to add an online introduction topic to the INTRODUCTION TOPICS MENU in addition to adding your application to the VM/IS panels. You should plan on performing both tasks at the same time.

A VM/Integrated System Online Introduction Topic

Your introduction topic should give a good introduction to your application. Among the topics you should include are:

- Overview
- Specific information about available facilities
- How to use the application.

Try to give examples wherever possible; you could even consider giving an example the user can work through, simulating actually using the application.

Remember that you need to add the introductory topic to the INTRODUCTION TOPICS MENU. Adding your topic is similar to the task of adding your application to the VM/IS panels. You should plan on performing both tasks at the same time.

VM/Integrated System Help Panels

It is recommended that you use existing help panels as a base to design help panels to accompany your application. Using this strategy, you can copy an existing help panel file, then overwrite the information to provide the necessary help information for your own application. Remember that you must supply your own component code for your files.

By copying an existing file, you can easily follow the existing help panel structure and format.

Try to keep your help panels short with clear information. A lot of information on a help panel intimidates a user and makes an existing problem worse. Remember that good help panels are an integral part of your application. Users judge the usability of the application according to the usefulness of the help panels.

For information on the type of information to include in your help panels, see "Online Help and Tutorial Information" on page 38.

The Application Program

When you finally get to the task of designing your application, you must take several things into account. For example, if you are using the facilities of another application, that application may have requirements on the structure of your application. In addition, each programming language uses a different structure. Some programming languages provide subroutines that save your programmers time in writing their code; implementing these subroutines requires you to use specific structures.

Customizing an Existing VM/Integrated System Product

Many supplied VM/IS products can be customized to meet requirements of your application, and you should customize them accordingly. Radical customization is not recommended, however, because the next VM/IS product migration procedure could well negate your changes or make the migration procedure more difficult for the customer.

Also, remember that changes you make to a product could well be changed again by a customer upon installation. If your application heavily counts on a specific customization, your application may not perform as you intended or may not function at all if your customer decides to customize the same product. Changes you make to other products should be well documented to ensure that a customer does not accidentally disable your application.

Tailoring the System for an Application

In designing your application, you may find that the application needs a special configuration of the system to work the way you intended it. This section discusses changes you can make to VM/IS to facilitate the operation of your application.

If customizing the files gets complicated, you may want to consider including a tool the customer can use to customize files. The tool could use one of or a combination of REXX, XEDIT, or ISPF. If you decide to supply such a tool, be sure to document its use.

Disk Layout

Migration is an important consideration when you design your application. The VM/IS system is refreshed frequently, allowing the customer to get enhanced versions of the products that make up the system.

When a customer installs a new release of VM/IS, the customer then ***migrates*** the older system to the newer system. The migration tool is designed to migrate only the system-owned volumes. Table 1 lists the system owned volumes.

Volume Name	Purpose/Contents
VMSRES	VM Resident Volume
VMPK01	Secondary pack containing base functions
VMPK02	Secondary pack containing base functions for 9332 DASD only
VMPK04	Optional source files
PROFPK	Optional packages
OPTPK1	Optional packages
OPTPK2	Optional packages
OPTPK3	Optional packages
OPTPK4	Optional packages
OPTPK5	Optional packages (used by customers with 9332 DASD only)

Table 1. The VM/IS System Volumes.

The VM/IS migration procedure allows you to place your application on any direct access storage device (DASD) volume where you can find sufficient space to define a minidisk. The migration procedure, when invoked, checks for non-VM/IS files on VM/IS DASD volumes and does not overwrite the application files.

VM/IS also allows you to add your own DASD volume where you can place your application. In this case, the migration procedure makes no changes at all to the contents of the added DASD volume. Either way, the customer can upgrade the VM/IS system without overwriting your application. Remember that it is your responsibility to ensure that your application runs correctly with the upleveled products after the VM/IS product migration procedure. For more information on DASD layouts and adding new DASD volumes, see *VM/IS Managing Your System.*

On the MAINT 326 minidisk, it is recommended that you place the panel tailoring files for adding your application to the VM/IS panels. Tell the customer to link and access this minidisk as CMS mode letter N.

You should supply another minidisk on which your application can be installed. In most cases, this is a MAINT-owned minidisk. This minidisk should not be one of those used by VM/IS. To obtain a list of minidisks that are used by VM/IS, log onto the MAINT user ID and issue the command:

```
DIRMAINT REVIEW
```

A file called MAINT DIRECT is sent to MAINT's reader. The MAINT DIRECT file contains a list of MAINT-owned minidisks.

Choose a minidisk address for your application that is not in the same range as the VM/IS minidisk addresses. This precaution ensures that the probability of VM/IS using your minidisk address in a future release is low. See "System Restrictions" on page 77 for the available addresses and filemodes.

Your application may not be the first non-IBM application installed at your customer's site. Even if the customer has a MAINT 326 minidisk and minidisks that can hold your application, the customer may have to enlarge the minidisks. You should give instructions on the way to enlarge the minidisks. See *VM/IS Managing Your System* for details on enlarging minidisks.

DMKRIO, DMKSNT, DMKSYS, and DMKFCB Assembler Files

If necessary, you can tailor the system files (DMKRIO, DMKSNT, and DMKSYS). In most cases, however, you should not need to do so.

You would need to tailor the DMKSYS file only if developing a monitoring application with a function similar to the Virtual Machine Monitor Analysis Program (***VMMAP***) function.

Avoid changing the DMKRIO file, although you may have to modify the file if your application requires a specific device that does not appear in the supplied file.

Another file that could be changed under certain circumstances is the DMKFCB file. This file needs changing only if, for reasons important to the application, the system printer must be exchanged for another printer. This procedure is not recommended.

If your application requires a saved segment, then the DMKSNT file must be changed. You should allow the customer to select the storage address (where in virtual storage the saved segment resides) and modify the file accordingly. This procedure avoids the possibility of a conflict in storage with other saved segments.

Saved segments are segments of reentrant code shared among many users. The use of saved segments results in better system response time, more efficient use of system storage, and reduced Input/Output (I/O) time, assuming that many users will be using your application simultaneously.

The system name table (DMKSNT) contains the layout of the saved segments. This layout is contained in the MAINT 295 disk in a file named DMKSNT ASSEMBLE. The DMKSNT file describes the following information to the system:

- What to copy when the segment is saved
- Where to store the copy on system-owned disk storage
- Where to load the copy into a virtual machine.

More than one saved segment may occupy the same area of main storage. If your application makes use of saved segments, the customer needs a system administrator with data processing (DP) skills to plan and set up the saved segments.

Note: Remember that not every customer has a system administrator with DP skills to install and maintain the system name table.

If you decide to use saved segments, you should refer the customer to the facilities provided by VM/IPF for updating the system name table. These facilities allow the customer to manipulate the system name table to install your application.

See *VM/IPF System Tailoring* for details about the System Name Table (DMKSNT).

You may want to use the SNTMAP EXEC to produce new DASD and MEMORY SNTMAPs to verify any changes you make to the DMKSNT file.

The SNTMAP EXEC is a DMKSNT mapping tool shipped as part of the base of VM/SP. SNTMAP resides on the MAINT 193 minidisk. The SNTMAP EXEC provides two listings:

- DASD SNTMAP

  This file contains a list of the saved segment names and direct access storage device (DASD) allocations arranged in order of occurrence on a pack.

- MEMORY SNTMAP

  This file contains a list of saved segment names and segment allocations arranged in order by memory location.

For more information on the SNTMAP EXEC, refer to the *VM/SP Installation Guide*.

The System Directory

Your application files will reside on one or more minidisks that users must access to use the application. There are three ways of arranging access to the minidisks:

- Make changes to each user's directory entry to link and access the minidisks each time the user logs on.
- Link and access the minidisks as part of your application invocation.
- Make changes to the system profile (SYSPROF) used by each user in logging on.

The second alternative is recommended unless the user will be using the application more than fifty percent of the time he is logged on.

If you elect to change the user's directory entry, pattern entries for your application after similar entries for existing applications.

If you link with the minidisks as part of your application's invocation, avoid using minidisk passwords to reduce the amount of administrative time required to change the passwords. In addition, you do not have to inform users of updates or change required passwords in your invocation EXEC. Instead, you should create additional directory links to provide access to minidisks.

In addition to tailoring individual directory entries, you may decide to use service machines in your application. Service machines are virtual machines dedicated to running particular applications. Using service machines can free user virtual machines, which the customer can use to run other applications. In addition, service machines can be used when several users must share data.

For example, the ***VM Batch Facility*** virtual machines are generated and controlled by a user ID dedicated to running jobs in batch mode (the BATCH user ID). The VM Batch Facility is particularly useful for jobs such as assemblies and compilations and for running large user programs. The customer can continue working on one virtual machine while time-consuming jobs can run in the VM Batch Facility virtual machine (BATCH).

Consider implementing "administrator" commands that can be sent to a disconnected service machine (by means of CP SMSG/IUCV support), so the administrator need not log on to the service machine for "normal" work. You should implement this facility in the same way that you implement communications with end-users. Generally, administrators can request some service machine functions that are not allowed for end-users.

Other Files

AUTOLOG1 is a disconnected virtual machine. VM/SP automatically autologs this virtual machine when you perform an Initial Program Load (IPL) of the VM/SP system. You can customize the SYSTEM PROFILE, which resides on the AUTOLOG1 user ID, to control the user IDs that are to be autologged at IPL time.

If you use disconnected service machines in your application, you should consider asking the customer to edit the SYSTEM PROFILE so that the disconnected service machines that your application uses are autologged whenever the customer performs the IPL on the system.

The customer can use the SHUTSERV EXEC to ensure that disconnected virtual machines are shut down before the system is shut down. When the customer runs this EXEC, the disconnected virtual and service machines listed in the SHUTSERV TABLE are shut down. The SHUTSERV EXEC and the SHUTSERV TABLE are located on the MAINT 19E minidisk.

If you use disconnected service machines in your application, you should consider asking the customer to edit the SHUTSERV TABLE to include the disconnected service machines that your application uses.

Several commands are used to shut down disconnected virtual machines. The correct commands to use depend on which disconnected service machines the customer wants to shut down. Make sure that you tell the customer to use the appropriate commands to shut down your application's disconnected service machines. Remember to tell the customer that the user ID beginning in column one of the SHUTSERV TABLE **must** be the same as the user ID used with the FORCE command later in the same line. In addition, the entire line must be in uppercase characters.

Internal Communications in Your Application

VM/IS gives you the ability to send data to and receive data from other user IDs (disconnected service machines used by another application) or the operating system. This facility is supplied by VM/SP, and is an internal communications facility.

The facilities supplied by VM/SP are:

- The Inter-User Communications Vehicle (IUCV)
- The Virtual Machine Communication Facility (VMCF).

It is recommended that you use IUCV rather than VMCF. It is beyond the scope of this book to explain the usage of IUCV facilities or of VMCF. To locate more information about IUCV and VMCF refer to *Developing an Application: Related Documentation*.

Inter-System Communication Applications

You can use Systems Network Architecture (***SNA***) and Binary Synchronous Communication (***BSC***) protocols to communicate with other systems through products like the Remote Spooling Communications Subsystem (RSCS) or the Virtual Telecommunications Access Method (VTAM).

In addition, VM/SP offers the Transparent Services Access Facility (TSAF) to allow you to easily communicate with another system running the same level of operating system (VM/SP).

Summary

To successfully design a VM/IS application, we recommend you do the following:

- Produce, as your objective, a set of detailed specifications from which your programmers can work.
- Design the file and data structures. Things to consider include:
  - The requirements and type of data used by your application as determined in your planning stages
  - Other applications with which your application may interact
  - Component codes and file naming conventions.
- Design the user interface. Your interface should contain the following characteristics:
  - Interactive and panel driven, but with an optional command-driven interface
  - Invoked by an EXEC
  - Similar in appearance to VM/IS menus and panels
  - Help panels provided for each panel as well as a tutorial to assist your customers in learning the use of your application
  - Error and informational messages to communicate with the customers
  - Program Function (PF) keys consistent with those used in VM/IS and defined for the customer on each panel
  - Panel flow organized in a task-oriented manner, proceeding from general tasks to specific tasks.
- Tailor the VM/Integrated System – Productivity Facility to include your application. This means creating and modifying files to create the redesigned menus.

- Include the following in your online introductory topic:
  - Overview
  - Information about available facilities
  - Instructions on using the application.
- Include the following types of information on your Help panels:
  - Explanations of each option on a menu
  - Format and type of information required for each input field
  - Tasks the customer can perform on a specific panel
  - A list of active PF keys.
- Consider the available subroutines when designing your application.
- Customize existing VM/IS products (if necessary) to improve or add function to your application.
- Include or optimize your application by tailoring the following:
  - DASD layout
  - Assembler files including DMKRIO, DMKSNT, and DMKSYS
  - The system directory file
  - Other files including the SYSTEM PROFILE.
- Design your application's internal communication. We recommend you consider using IUCV (Inter-User Communications Vehicle) for your internal communications.
- Design your application's inter-system communication using products included in VM/IS.

Chapter 5. Coding a VM/Integrated System Application

The coding task involves implementing a solution to the design specifications you set in the design stage. VM/IS contains many program development tools and languages for you to code your applications.

Introduction to Coding

Although coding the application may seem like the culmination of the task of building an application, it is really only the next step in developing your application. You started with an idea for an application, established objectives, and designed the specifications. Now you assemble the code that becomes the application. There remain the tasks of testing, packaging, and documenting. Each task is equally important.

If your specifications are adequate, the task of coding the application is made easier. All the programmers must do is supply code that performs the functions your specifications detail. Their task is implementation rather than planning or designing.

The Application Program

This section discusses the actual task of coding your application program. Topics discussed include:

- Using the application development languages
- Using predefined routine and utility languages
- Coding with two or more programming languages
- Using the system languages and interfaces.

The Application Development Languages

You can program your application in any one of several languages including:

- Assembler, contained in the VM/IS BASE package
- VS COBOL II, contained in the Transaction Support (TXN) package
- REXX (Restructured Extended Executor language), contained in the VM/IS BASE package
- VS FORTRAN, contained in the Engineering/Scientific Program Development Support (E/SPDS) package
- Other languages that run under VM/SP. Several of these languages can be ordered under the VM/SP System Offering and could include:
  - APL2
  - IBM BASIC/VM
  - Pascal/VS.

Of course, each language has it's advantages and strong points. The following sections discuss this in more detail.

Assembler Language

You can use the assembler language to perform any programming function, but it is a difficult language with which to work. It is one step away from machine language, and each instruction performs one specific operation. In addition, because there are so many higher-level languages available, not many people are available who can program well in assembler language.

You can use assembler language to perform specific VM functions and take advantage of specific devices. In other languages, this task is more difficult. Assembler is efficient and programs run fast, but your programmer cost, both to write the code and to maintain the code, is higher than with another language.

It is recommended that you use the assembler language for device-specific or system-function-specific subroutines that can be called by your high level language mainline programs, as well as for routines critical to performance.

REXX

REXX (Restructured Extended Executor Language) is the System Product Interpreter language. It is an interpretive system language that is easy to learn and easy to use. The language is best used to issue CP and CMS commands, but can also be used for other functions. REXX cannot be compiled, but it does include tracing facilities that make programs easy to debug. It is especially suited for use in setting up the user environment prior to calling a high-level language for writing XEDIT macros and ISPF dialog functions.

VS COBOL II

VS COBOL II is the version of COBOL that runs on VM/IS. COBOL is the most widely used application programming language in the world because of it's power in handling business data processing needs. Programs written in COBOL are geared to handle large volumes of data, manipulate the data and produce a variety of reports and files. COBOL is also flexible enough to handle a wide range of data processing needs from overnight payroll processing to online database inquiries. The reason for COBOL's popularity is that the COBOL statements resemble English sentences, thus making the COBOL program very easy to read and just as easy to maintain.

VS FORTRAN

VS FORTRAN must also be compiled and link-edited to run any sections of code. It is a powerful mathematically based language particularly suited to scientific applications. Its strong point is floating point calculations, but it is not as good at string manipulation. Consider using it if you are building a "number-crunching" application. While VS FORTRAN does not require each line to be numbered as does IBM BASIC/VM, it does require that any label appear in the form of a number, rather than a character string. This requirement could make the language more difficult to use, because a label cannot relate to the task performed by the section of code it references. Programmers using VS FORTRAN should be skilled in its use.

Other Languages

If you have programmers skilled in languages other than those included in VM/IS (for example, APL2, IBM BASIC/VM, Pascal/VS, or PL/I), you might want to consider coding your application in that language.

You can, of course, install other languages on VM/IS to code your application. Remember that the language must run on a VM system. If a customer does not have that language installed on his system, you must supply precompiled modules. In addition, servicing your application could be more difficult.

There are certain advantages in using a different language. For example, if you elect to use PL/I, remember that VM/IS includes the ***PL/I Transient Library*** and several optional packages contain the ***PL/I Resident Library***. You could code your application to take advantage of this option, thereby saving both time and work.

The following sections discuss some of the languages available through the VM/SP System Offering:

APL2: APL2 is a high-level language well suited for problem solving, data analysis, and applications. You might decide to use APL2 if developing an application for a business or manufacturing environment. It is a language well suited to matrix manipulation, because the language uses matrices as part of its structure.

Be aware that when calling subroutines written in other languages you should check that your code gives you the hoped for result. In addition, you need a special keyboard to program in APL2 because it uses special symbols. Because it is interpretive, it is easier to debug. Programmers using APL2 should be highly skilled to get the most out of the language.

IBM BASIC/VM: IBM BASIC/VM is a versatile language used by a large number of people because it is easy to learn and use. It is a general language that can solve problems in a large number of areas and is especially good at string manipulation. The language does not have as much function as other languages. One disadvantage is its inability to use labels as pointers, instead requiring each line to be numbered.

IBM BASIC/VM is interpretive, although it can also be compiled and link-edited. It is easy to debug; a programmer can run sections of code without compiling. For the same reasons, however, even after compiling, an application tends to run more slowly than an application coded in another language and is more likely to contain undiscovered errors. IBM BASIC/VM is best suited to small, relatively simple applications. If you are a small site, or have limited skills, you can use IBM BASIC/VM to build an acceptable application.

Pascal/VS: Pascal/VS is another language that is easy to learn, although it is not as easy to master as IBM BASIC/VM. It is a highly structured language, suitable for writing both system programs and applications. Any code must be compiled and link-edited before you can run it to check its function.

Predefined Routine and Utility Languages

VM/IS supplies you with a large number of predefined routines in a variety of programming languages. In many cases, you can call these routines to perform a specific function rather than actually writing the code to perform that function (for example, VS FORTRAN includes predefined subroutines). In addition, a number of subroutine libraries are included as part of VM/IS. VM/IS includes the PL/I Resident Library and the PL/I Transient Library as well as several CMS macros you can use as part of your application.

You should familiarize yourself with the available subroutines before beginning to code your application.

Using Two or More Programming Languages

All languages supplied with VM/IS have the capability of calling subroutines written in other languages. Some languages handle this task better than others, however. Whichever language you decide to use for your coding, there are specific protocols to follow in including subroutines. Ensure that you are familiar with the protocol before trying to use these subroutines.

Investigate all possible subroutines before deciding on a specific subroutine, because you may find another subroutine written in another language that performs the required task more efficiently.

In addition, other applications often have tools to facilitate their use. You can take advantage of these tools by utilizing the command sets supplied by the applications. For example, you can interact with an SQL/DS data base using facilities supplied by SQL/DS, or you can use facilities supplied by other products, like ***QMF/VM***.

The System Languages and Interfaces

You could decide to extensively use the System Product Interpreter, ***REXX*** (this is the short form for the Restructured Extended Executor Language) in the development of your application. REXX is easy to use and is capable of a variety of function. In addition, REXX is an interpretive language, so you can easily run and debug your code. REXX includes an interface to SQL/DS to enable you to use the functions of SQL/DS in your REXX applications. For more information about using REXX, see "REXX" on page 50.

In addition, you have the ability to use CMS commands in your application. Plan to take advantage of these facilities as much as possible to save coding time and expense. If you choose to use one of the available facilities, however, remember that your application then becomes dependent on the facility you use. If that application (or facility) is upgraded to a newer level during migration, your application may no longer work properly.

VM/SP includes a parsing facility that checks all operands, options, and keywords included in commands. Since the syntax definitions are stored in separate files, you can create files to have VM/SP check the syntax of commands used by your application. Then you do not have to include syntax checking as part of your application, and commands used by your application can be easily translated into other languages.

The Development Tools of VM/Integrated System

VM/IS supplies you with tools to assist you in the development of your application. Included are tools to assist you in coding, testing, scheduling, and controlling your project.

Program Development Tools

The most important program development tool supplied with VM/IS is ***ISPF/PDF***. You can use ISPF/PDF to:

- Create and edit programs using a variety of programming languages
- Store and retrieve your programs using libraries you create
- Compile and execute your programs through interactive or batch facilities
- Perform general maintenance to your program libraries.

You can develop and test your code using ISPF/PDF. You can also use it to create ISPF dialogs.

Project Management and Control Tools

As your project develops, you or your programmers may discover problems that you must correct with design changes. A change may involve only a small amount of code, or it may mean reorganizing an entire function.

A change can affect your project in predictable ways. For example, an error condition can be caused by a specific section of code that does not perform according to specifications. This type of error can be fixed by recoding the section that caused the error.

Changes can affect your project in other ways. Someone may recode a small section of code to fix an error condition without realizing that yet another section of code has been affected by the change. This change thereby causes another problem.

Larger problems could necessitate recoding an entire function. This could affect the entire application, including any documentation that is being developed.

You need a way of handling these changes to ensure that everyone affected is informed. You also want to keep a record of changes or problems to ensure that problems have been solved and changes implemented.

ISPF/PDF includes a library management system to assist you in controlling your project. In addition, you can use the CMS UPDATE command to modify and update source files while keeping a log of the changes.

Sometimes changes can affect the application development to the extent that you must change your schedule. See "Managing Your Project" on page 29 for information about schedules and managing your project. When changing your schedule, ensure that you inform all affected people.

Text Editors

Whatever language you decide to utilize, your programmers must enter their code on the system. Your testers must document their testing strategy and the results of the tests. Your writers will have information they must enter in the process of documenting your application. In addition, you will have information you have to enter or modify during all phases of the project.

VM/IS supplies tools to assist you, your programmers, your testers, and your writers. The major tool is the System Product Editor (XEDIT). You and your people can use XEDIT to enter most of the information or code they write. Or you can use the ISPF/PDF editor. In addition, ISPF gives you the facility to use XEDIT as do other products.

Each editor gives you facilities to make easier the task of entering or changing information. Familiarize yourself with the available editors and their features. For example, XEDIT has excellent macro capabilities, allowing you to control it by a REXX program. Thus, XEDIT is a very powerful utility.

You may decide to use XEDIT through ISPF/PDF to gain the advantages of both. Or you may decide to use only XEDIT or only ISPF/PDF with its own editor. Whichever you decide, ensure that your programmers are comfortable with the decision and that they are knowledgeable with the editor. You may want to run one or

more seminars to ensure that your people can use the chosen editor to its full advantage.

The User Interface

When coding your user interface, you can develop a menu-driven interface or a command-driven interface. It is recommended that you use a menu-driven interface to make your application easier to use and to blend into VM/IS.

When developing a menu-driven interface, you have the choice of using ISPF or ***GDDM*** as your dialog manager. You should use ISPF as your dialog manager for the following reasons:

- Your application blends into VM/IS, because ISPF is used by VM/IS.
- You can save time and resource because ISPF is started and running in a normal VM/IS environment before your application is invoked.

If your interface uses or interacts with graphics, you may decide to use GDDM as your dialog manager. Still another alternative involves using ISPF as your dialog manager and calling GDDM to interact with any graphics that your interface requires.

Customizing the VM/Integrated System – Productivity Facility

To make your application accessible by your customers, it is recommended that you customize the VM/IS – Productivity Facility to include your application. You must create and include as part of your package the files necessary to the tailoring process. In addition, as part of your project, you should add an online introduction topic and VM/IS help panels.

This section discusses the following tasks:

- Adding your application to the VM/IS home menus
- Adding a VM/IS online introduction topic
- Adding VM/IS help panels.

Because you actually perform the task of customizing the VM/IS – Productivity Facility in a single operation, this book presents the above tasks as a single topic.

In your planning and designing phases, you established how you want your application to appear on the VM/IS home menus. You probably designed the final menus. In addition, you designed your online introductory topic, and your VM/IS help panels.

As much as possible, you should base your new panels on existing panels to lower the possibility of error. For example, if you copy an existing panel file, then replace the information in the copied file with your own information, your file automatically meets size requirements. This is an important consideration if parts of your application are to be translated into another language.

Also during the planning and designing phases, you investigated the concept of tailoring the VM/IS – Productivity Facility to understand the procedure and complexity of the task. In this phase, you implement your plans and designs.

Tailoring the VM/IS home menus is a two part process. The first part creates the new files which add your application to the menus. At this point, you can test your implementation. As soon as you are satisfied with your tailoring, you can create and edit files to package your tailoring files. Then, when you run the PFSETUP EXEC, the system menus change according to your specifications.

The *VM/IS Tailoring Your Menus* manual shows you how to add your application to the menus and test your changes as well as how to package the files. Follow the step-by-step instructions to create the files to add your application to the VM/IS home menus.

Remember that you can tailor the PROFS Main Menu, the PROFS Subset Menu, and the PROFS help panels as well as the VM/IS – Productivity Facility panels.

Note: You can tailor the PROFS help panels only if you have the Document Composition Facility (***DCF***) installed.

Tailoring Existing VM/Integrated System Products

Often, to implement a specific function for your application, you find that you must change an existing VM/IS product by tailoring the product or by adding function. For example, you may find that your application must format a report in a format not supported by DCF as supplied. You may decide that rather than redesigning your report format, you can produce the desired result by adding tags to the ***GML*** starter set.

You can tailor several products, including DCF to augment your own application. Ensure that you meet all requirements for the product. Try to customize the product so that the migration process does not negate your changes.

Summary

To successfully code a VM/IS application, you should:

- Use one of the application development languages to code your application. Languages you can use included with VM/IS are:
  - Assembler
  - REXX (Restructured Extended Executor Language)
  - RXSQL (VM/SP Interpreter Interface to SQL/DS)
  - VS COBOL II
  - VS FORTRAN Version 2
  - Other languages that run under VM/SP and are available as part of the VM/SP System Offering. In this case, either the customer must be prepared to supply the language as a prerequisite to installing the application, or you must supply the application already compiled and link edited. Languages include:
    - APL2
    - IBM BASIC/VM
    - Pascal/VS
    - PL/I.

  Remember that you can code most of your application using one language, and call subroutines that use other languages.

- Use the tools VM/IS includes to assist you in coding your application. These tools include:
  - Program development tools
  - Project management and control tools
  - Text editors.
- Code the user interface using either ISPF or GDDM dialogs.
- Customize the VM/IS – Productivity Facility by adding:
  - Your application to the VM/IS home menus
  - An online introductory topic
  - Help panels for your application.
- Tailor existing VM/IS products if necessary to provide function for your application that otherwise might not exist.

Chapter 6. Testing Your VM/Integrated System Application

The purpose of this chapter is to provide a brief discussion of the importance of software testing, and to point you towards some of the testing tools and monitor programs available on VM/IS. You should test your application extensively, both for correct function and for proper interaction with the rest of the VM/IS system.

Introduction to Testing

You should always start testing with the assumption that the program contains errors. The purpose of testing is not to prove that no errors exist, but rather to find as many errors as possible.

There are many phases of testing, each intended to focus on certain kinds of errors. Your goal at each phase is to make the application fail.

It is important that you decide what is an error and what is not. To decide, use the following guidelines:

- Ensure that the specifications for the application are complete, consistent, and measurable, and say exactly what your application should do.
- Define specific standards to distinguish between errors and acceptable responses.
- Measure all quality characteristics in the specifications. For example, if your specifications contain ease-of-use and response-time targets, measure your application against these targets.

These principles apply to all stages of testing. If you have well-defined specifications, standards and measurement techniques, you have then defined exactly what is an error and what is not.

Testing then becomes a process of performing the measurements and reporting the deviations between the program and the specifications. You should keep detailed records of deviations, type, severity, origin, and method of discovery.

A final important principle of testing is that you should not test your own programs. Remember that the purpose of testing is to find as many errors as possible. It is difficult to remain objective when testing your own program. Testing, therefore, should be performed by an objective, independent party.

Testing Your Application

Testing should not be a last-minute activity. Because you can correct errors more easily in the early stages of development, testing should begin early in the development cycle. You can even start testing before you start coding.

Do not underestimate the importance of testing as early as possible. The earlier you obtain feedback, the more easily and inexpensively you can make corrections. Remember that the ultimate test is the use of the software by customers, when corrective action is expensive and sometimes too late.

During the planning stage for your application, you may find it helpful to discuss your plan with others. While this discussion is not a formal test, you are testing

your ideas to see whether they make sense. Weaknesses sometimes show up when another person looks at the idea from a different point of view.

During the design stage, you draw flow charts and logic diagrams, and write specifications. You may even write ***pseudo-code***. All of this can be tested. You should hold formal meetings to let others examine your logic. Errors are often discovered when a group of people examine and discuss the logic flow. Remember that the further into your application development you progress, the more difficult it becomes to repair problems.

In addition, design the application with testing in mind. You can design the code with checkpoints to enable testers to interrupt and resume the execution of the code as necessary. This method saves time, because the testers do not have to reinitialize the application for each test case.

You and others can test your code at the same time as you actually code the application. As soon as you finish a section of code, you can compile that section (if necessary) and test it. As you finish more sections of code, you can begin to put them together to form modules, which again can be tested.

Remember that errors removed early, when you test a section of code, save time later. In fact, the longer you wait to test your code, the more interaction there is between sections of code, thereby making it harder to locate what may be small errors.

VM/IS supplies tools, debugging aids, and guidance to help you test your application. These aids vary depending on the language you use to code your application and whether or not your application interacts with other applications. For example, if you code your application using VS FORTRAN, you can test and debug your application interactively.

Executing Your Application

As soon as you finish coding, you should test the entire application. First test the application's functions on an individual basis. Test that each function performs as it should and produces the correct output.

If a function interacts with another application, ensure that the interaction does not affect the operation of the other application. You should check the status or condition of the secondary application and its data before any interaction. Then invoke your application's function and recheck the status of the secondary application to ensure that it has not been adversely affected.

After testing your application on a function-by-function basis, test the application as a whole, with all functions interacting. Although you may be able to use some of the same tests, be sure to design additional tests to ensure that there is no change even with all functions interrelating.

Remember that you cannot test too much. If you do not find a problem in the application during testing, one or more of your customers are sure to find it later. The cost of fixing the problem at that point is much greater.

It is suggested that you develop a test plan containing actual test cases and data that can be verified. Do not deviate from your test plan without documenting the change. In addition, document your results. If anyone wants to know that the application was properly tested, complete documentation is essential.

For more information on test plans, see "Creating Test Data" on page 59.

Testing for Performance

You may want to test your application for performance. If your application is large, and if many people use it at any given time, performance is an important consideration. Performance is even more important if your application is one of several applications installed on a system. You don't want your application to be the one that slows down total system response time.

Testing for performance should be an ongoing task. If you test for performance after completing the coding and discover that the application has poor performance, you may have to make major design changes to improve it.

Some aspects of performance can be improved easily without major changes. If, when testing the entire application, you discover that performance has decreased noticeably from when you tested the individual functions, there may be a problem in the way the functions interact. In this case, testing for performance may have uncovered a problem that would not have shown up in any other way.

Tools available to help you test for performance include VMMAP and VM RTM.

Using Debugging Aids and Tools

As noted previously, debugging aids and tools vary extensively depending on the language in which you code your application.

Most languages offer error messages and codes to assist you in debugging your application. The language documentation explains error messages and often gives possible courses of action to fix the problem. In addition, several VM/IS products such as ISPF and VM/SP offer tracing facilities to help you debug your application.

The applications with which you interact usually have error reporting facilities as well. If your application interacts with another application, and this interaction generates an error message, you must discover which application generated the message. Check the *VM/IS Learning to Use Your System: Error and Information Messages* for a list of all message prefixes that could be generated by products in VM/IS. You can easily trace the product that generated the message and deal with it.

Creating Test Data

You should approach the task of testing your application in an organized manner. Document your intentions, your data, your expected results, and the skill level of the testers. In other words, design test cases or scenarios using real and verifiable data. If you cannot verify that the results of a test are correct, there is no point in performing the test.

Try to put your test cases in real life terms. For example, when testing the entire application, try to imagine yourself in the role of the person using it.

The tasks differ depending on whether you are testing a small section of code, an entire function, or even the entire application. The larger the section you test, the more complicated becomes the testing task.

Test Case Design

In designing your test data, use the initial specifications, flow charts, and logic diagrams to determine how the application should react in any given case. If you don't know what the application should do, you will have trouble judging the results of the test.

Design test cases to check that the code performs the task it was designed to perform. These test cases use data that falls within the limits of the application. You should also create test cases with data that falls outside the limits of the application. This type of test confirms that the previous tests (using valid data) show accurate results and test the recovery capability of the application.

You should perform both functional testing and usability testing. Functional testing ensures that the application meets specifications. Usability testing ensures that your intended customers are able to use the application to perform specific tasks.

Functional Testing

When performing functional testing, use experienced testers who both understand testing and understand the system on which the application is installed. Such persons are most likely to find errors, and you can design test data for them that is beyond the scope of normal end-users.

Usability Testing

For usability testing, try to use testers with the same skill level as the people for whom the application is designed. For example, if your application is used by secretaries, it would be good to have secretaries test it. Design test cases that simulate tasks performed by your intended audience.

Testing the Documentation

In addition, when testing the application, remember to test the supplied documentation. The application may perform as stipulated, but if the manuals give incorrect, incomplete, or misleading information, the customer may not be able to perform the task. The supplied documentation should include not only the hard-copy manuals, but also the Memo to Users (included as a print file as part of the packaged application), the online help panels, and the tutorial (if included). The documentation should work with the application and appear to be part of it.

Testing the documentation should occur early enough in your testing that any necessary changes can be implemented. For example, if a manual has already been published, and you discover a major error, you may have to scrap the published manuals, correct the error, and reprint the manual.

As with testing the application, try to test the documentation using testers with the same skill level as the people for whom the application is designed.

Testing the System

It is suggested that you actually install the application on VM/IS, and tailor the system as it would be tailored for a customer. With the application installed on VM/IS, you can then test it to ensure that your application interacts correctly with all other installed products and packages. This testing is especially important if your application uses the resources of other VM/IS applications.

If you have spent enough time planning, designing, coding, and testing, you should find very few problems when performing this final series of tests, and any problems that you do find should not be large.

An important part of testing the system involves testing other applications installed on the system to ensure that they are not affected by the addition of your application. Ensure that there are no conflicts in naming conventions. That is, test that, when other applications call specific modules, the correct module is accessed. Also check that your application accesses the correct modules. If you have tailored other applications to meet a specific need of your own application, ensure that the tailored applications still function as supplied. Ensure that system performance has not been affected by the addition of your application.

The best strategy for performing this type of test is to use the system the way a customer would use it. Check that the system functions normally and watch for abnormalities.

Summary

The purpose of testing is not to prove that no errors exist, but rather to find as many errors as possible. Start testing with the assumption that the program contains errors. To successfully test a VM/IS application, you should do the following:

- Define an error. You must have complete and measurable specifications, you must define exactly what constitutes an error, and you must measure all of the measurable specifications.
- Begin testing early. You can test parts of your application while you are planning, designing, coding, documenting, and packaging.
- When testing your application, start by testing each function individually. Test the application as a whole only when you are satisfied that each function performs as defined in the specifications.
- Document your test plan, the data you intend to use for testing, and the results you obtain. Do not deviate from your test plan without documenting your actions.
- Test for performance as part of your test procedure. Performance testing may uncover other problems.
- Use tools and debugging aids, such as:
  - Messages and codes issued by the languages you use to code the application
  - Tracing facilities
  - Messages and codes issued by those products with which your application interacts
  - The *VM/IS Learning to Use Your System: Error and Information Messages* manual.
- Use real and verifiable data for your test cases. Document the data you intend to use for testing, and the type of tester (usually the type of end-user that uses the completed application).
- Use not only data that you know should work in the application, but also data that you know should not work.
- Test the entire application for usability. Use testers with the same level of skill as the intended users.

- Test the documentation as a normal part of your testing procedure.
- Test the entire system with the application installed. Test not only the application itself, but also the other products installed to make sure that all parts of the system interact without errors.

Chapter 7. Packaging a VM/Integrated System Application

After you finish developing your application and creating the files to add your application to VM/IS, you must have a way of distributing your application. Because VM/IS is distributed by tape, you should distribute your application using the same method.

You should put a copy of your application onto a tape and supply documentation to install, customize, maintain, and use the application. The procedure and format used to put the application on tape is called packaging, and the tape including documentation is called the package. The following sections discuss the format and contents of your package tape in more detail.

A Discussion of Packaging Tasks

In packaging your application, remember that you are distributing to customers who may not have a system programmer on staff. You must supply your application in a format that is easily installed. In addition, you must supply tools to facilitate the installation and documentation to guide and assist the customer. (For information on developing documentation, see Chapter 8, "Documenting a VM/Integrated System Application" on page 81.)

If the customer has to do any setup before he or she can install the application, make sure you document the information. If your application requires its own service or administrative machines, supply the information necessary for setting up the user IDs. This information should already be available, because you developed it as part of the design phase of your application. (See Chapter 4, "Designing a VM/Integrated System Application" on page 33.)

Other concepts to take into consideration when packaging your application are:

- Automated installation

  Try to avoid manual steps unless they are necessary.

- Minimal skills required to install

  Remember that the person performing the installation may not have system programming skills.

- Executable modules pregenerated

  Generating and link editing during installation can greatly increase the time required to install the application. Aim to include all executable modules on the package tape. If necessary, you can include two installation paths, a fast path using pregenerated modules, and a longer path that performs all link edits and generations. If you decide to have two paths, your Installation EXEC should prompt the customer to select a path, with the fast path being the default.

- Minimal tailoring

  If the application must be tailored during installation, give two paths, one allowing the customer to accept all defaults, and the other giving prompts to assist in the tailoring process. Give a prompt to allow the customer to choose the path.

- Automated verification

  If possible, make the verification process automatic. It should execute enough functions to ensure that the application has been installed successfully and is operational.

- Separate features.

  If your application has multiple features that can be ordered and installed separately, package each feature individually. Use a common installation EXEC for the application and its features, and a common verification EXEC to verify the installation.

The Package Tape Format

Your application may have one or more features that can be installed in addition to the application itself. If such is the case, you should package files for each feature separately. You can differentiate between the actual application and one of its features by adding or changing the last two digits of the filetype for the product identifier file on your package tape.

You should put the application and all its features on a single tape. Use the tape layout guidelines outlined in this section for your application. Then use the same guidelines to place each feature on the tape following your application.

Your customer uses the ***INSTFPP EXEC*** to install your application. The INSTFPP EXEC is located on the MAINT 193 minidisk and is included in each release of VM/IS. For the INSTFPP EXEC to properly install your application, you must adhere to a particular format when you package your application.

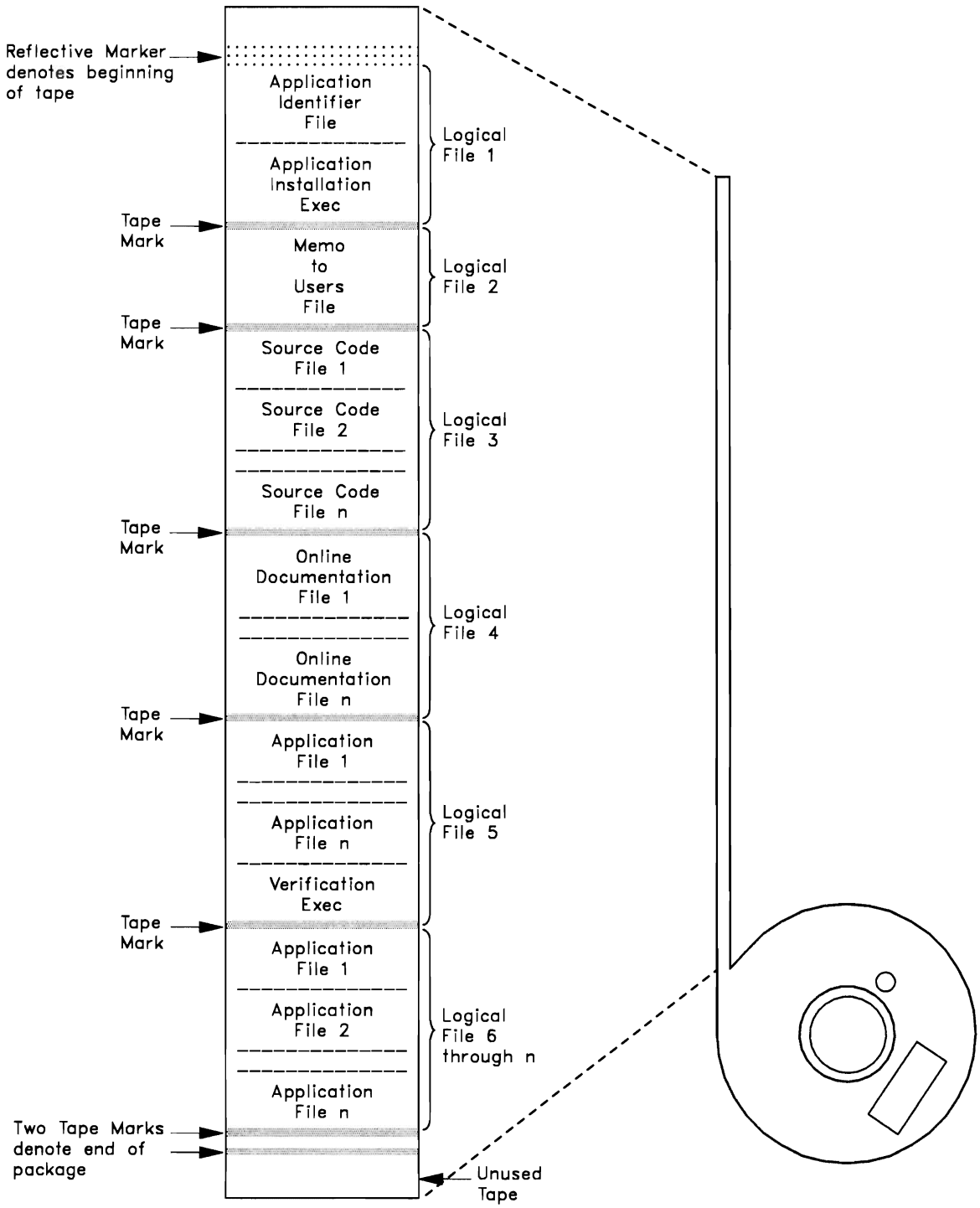


Figure 10. Package Tape Contents and Format

Your tape format consists of five or more logical tape files[3] for each application or feature. You must place a single tape mark after each logical tape file, and two tape marks at the end of the package (that is, at the end of the last logical file). See Figure 10 for an illustration of the tape layout.

Build the tape using the command VMFPLC2. See "Building the Package Tape" on page 75 for more information on the VMFPLC2 command and its use.

Logical Tape File Number	Logical Tape File Contents
Logical file 1	Application Identifier File Application Installation EXEC
Logical file 2	Memo to Users File
Logical file 3	Source Code Files (if applicable)
Logical file 4	Online Documentation (if applicable)
Logical file 5	Application Files (include Verification EXEC)
Logical file 6 through n (optional)	More Application Files (if applicable)

Table 2. Logical Tape Files.

Each of the logical tape files shown in Table 2 is discussed below:

1. Logical Tape File 1

   The first logical tape file must contain two CMS files: the application identifier file and the application installation EXEC. It cannot contain any other files.

   - The Application Identifier File

     The application identifier file naming convention uses the following format:

     Innnnnnn 0vrmbbii

     In the filename, *I* is constant (it stands for *install*) and *nnnnnnn* represents your application number. The application number may be any seven-character alphanumeric string. Your application number may not match any other product number that already exists. To make your application number unique, consider including your component code as part of the number.

     In the filetype, *0* (zero) is constant. The *v*, *r*, and *m*, are numbers representing the major version, release, and modification levels of your application. If your application is referred to only by release and modification level, the default version number is *1*. If the version, release, and modification levels do not apply to your application, you can set these values to zero. The *bb* represents the number of logical tape files on your application tape. The INSTFPP EXEC uses this number for controlling tape positioning during the installation process. The *ii* represents the alphanumeric identifi-

[3] A logical tape file is everything on the tape from the starting point of the tape to the point at which a tape mark is encountered, or subsequently, everything on the tape from that tape mark up to and including the next tape mark. A logical tape file could consist of any number of actual application files. These files may be grouped together because they are related to each other in some way.

cation code that identifies individual features of a product. If your application has no related features, omit these two digits.

For example, the accounting package example used in "Tailoring the VM/Integrated System Menus" may have an identifier file called `I3927A01 012005`. In this case, the application has no additional features, so the filetype contains only six characters. For information on the contents of the application identifier file, see "The Application Identifier File" on page 70.

- The Application Installation EXEC.

  The application installation EXEC uses the following naming convention:

  Innnnnnn EXEC

  In the filename, *I* is constant (install) and *nnnnnnn* is your application number. This is the application installation EXEC provided by you, the application developer. For more detail constructing this EXEC, see "The Installation Routine" on page 70.

Logical file 1 ends with a single tape mark.

2. Logical Tape File 2

   The second logical tape file must contain the Memo to Users file, which uses the following naming convention:

   Innnnnnn MEMOii

   In the filename, *I* is constant (install) and *nnnnnnn* is your application number. In the filetype, *MEMO* is constant and *ii* represents the alphanumeric identification code that identifies individual features of a product. If your application has no related features, omit these two digits. For more information on the contents of the Memo to Users, see "The Memo to Users" on page 87.

   Logical file 2 ends with a single tape mark.

3. Logical Tape File 3

   The third logical tape file contains source code files (any nonexecutable code). Include only those source files needed for installation or service of your application.

   If no source code files exist, then the contents of the next logical tape file will be placed here, thus moving all the logical tape files up one file.

   Logical file 3 ends with a single tape mark.

4. Logical Tape File 4

   The fourth logical tape file contains online documentation.

   If you do not provide any online documentation, then the contents of the next logical tape file will be placed here, thus moving all the logical tape files up one file.

   Logical file 4 ends with a single tape mark.

5. Logical Tape File 5

   The fifth logical tape file contains your application files and can contain the verification EXEC (if your application includes an automatic verification process). While you may decide to package your verification EXEC in this logical tape file, you could instead choose to package it in one of the remaining logical tape files (6 through n).

The verification EXEC uses the following naming convention:

Vnnnnnnn EXEC

In the filename, *V* is constant (verification) and *nnnnnnn* is your application number. For information about the verification EXEC, see "The Verification Routine" on page 74.

The fifth logical file also includes those files that add your application to VM/IS (for example, the panel tailoring files). Your installation EXEC must install these from the tape. While the fifth logical tape can contain all your application files, we suggest that, for ease of installation you create a separate logical tape file to contain the files for each minidisk.

Logical file 5 ends with a single tape mark unless it is the last logical file in the package. The last logical file must end with two tape marks.

6. Logical Tape Files 6 through n

   These logical files contain additional application files. You can divide your application files into as many logical files as you wish. We suggest that, for ease of installation, you create a separate logical tape file to contain the files for each minidisk.

   Logical files 6 through n end with a single tape mark, up to but not including the last logical file. The last logical file in the package must end with two tape marks.

You should follow the above tape packaging format for your VM/IS application. You can deviate from this format for logical files 3 through n, provided your application installation EXEC manages the tape as specified in "The Installation Routine" on page 70.

Suggestions

In addition to files required to install, execute, and service your application, you can ship other files to simplify the installation, the customizing, or the use of your application:

- PROFILE EXECs

  Applications that have their own user IDs can include PROFILE EXECs with the application files. PROFILE EXECs contain any setup required by user IDs or virtual machines when they are logged on.

- Panels for tailoring or customizing the application
- Panels to use the application
- Sample directory entries.

The VM/SP Installation Process

The application installation EXEC that you develop to install your application tape must interface with the INSTFPP EXEC, which is shipped as part of VM/IS on MAINT's 193 minidisk. All VM/IS applications are installed using the MAINT user ID.

The INSTFPP EXEC either displays XEDIT panels to get required information or accepts parameters with the INSTFPP command. You can find details on the INSTFPP command and its parameters in the *VM/SP Installation Guide*, or enter:

```
help instfpp
```

on VM/IS to see the online help information for the command.

The INSTFPP EXEC:

- Creates two temporary minidisks at virtual addresses 002 and 001 and accesses them as A and C respectively.
- Accesses the MAINT 191 minidisk as D, and the MAINT 319 as P.
- Loads the first two logical tape files of your application. Your application tape remains positioned at the beginning of logical tape file 3.
- Gets the first record of your application identifier file and issues a prompt.

  For example, the prompt may look like this:

```
3927A01  ACCOUNTING PACKAGE
Do you wish to install this product now (yes or no)?
```

  Note: The customer can also invoke INSTFPP with the NOPROMPT option to suppress the prompt being displayed.

- Prints your application's Memo to Users file.

  Note: The customer can also invoke INSTFPP with the NOMEMO option to suppress printing the Memo to Users.

- Executes your application installation EXEC, passing it two arguments (*YES* and the application identification code). The first argument tells your installation EXEC to bypass the user prompt (if your EXEC includes such a prompt), and the second argument handles multiple features within the application. See "The Installation Routine" on page 70 for more information.

  When your application is installed, control is passed back to the INSTFPP EXEC.

- Copies the Memo to Users from temporary minidisk **C** to the MAINT 319 minidisk.
- Checks the return code and updates the PROD LEVEL file on the MAINT 319 minidisk to indicate the results of each application installation. See "The Installation Routine" for more information on the meaning of the INSTFPP EXEC return codes.

The Application Identifier File

Your installation EXEC uses the application identifier file to identify the application to the person who is performing the installation.

This file must contain a minimum of two records and may contain additional records. The records contain the following information in the following format:

- Record one:

  Product number: Your application number, starting in column one

  Product name: The full name of your application, starting in column ten.

- Record two:

  Version: VER followed by the version number, starting in column one

  Release: REL followed by the release number, starting in column ten

  Modification: MOD followed by the modification number, starting in column twenty

See Figure 11 for a sample application identifier file.

Note: Figure 11 includes a scale to indicate the proper positioning of the record content. The scale is for illustrative purposes only, and is not part of the file.

```
|.....1.....2.....3.....4.....5.....6.....
3927A01  ACCOUNTING PACKAGE
VER 1    REL 1     MOD 0
```

Figure 11. A Sample Application Identifier file

The Installation Routine

The application installation EXEC is the EXEC you create and supply on the first logical file of your application tape. This EXEC loads your application files from the remaining logical tape files.

Use REXX (the System Product Interpreter Language) to write your application installation EXEC.

Design your EXEC assuming that the customer has already added required entries to the VM/IS BASE as instructed in your application's preinstallation documentation. Required entries may include:

- Directory entries defined and online
- Saved segment entries defined in the DMKSNT file and generated in the CP nucleus
- Entries for the DMKRIO and DMKSYS files.

Your application installation EXEC should do the following:

1. Be executed by the INSTFPP EXEC from the MAINT user ID where virtual storage is defined to be from 12M to 16M, and CMS is loaded. INSTFPP is on MAINT's 193 minidisk, which is accessed as **T**.

Note: If the customer elects to print the Memo to Users without installing the application, virtual storage can be defined to be less than 12M.

2. Read the two arguments passed by the INSTFPP EXEC (*YES* and the application identification code). Before invoking your installation EXEC, INSTFPP asks the customer if he or she wishes to install the product. Your EXEC may contain the same question, but it should not display the question if the customer has already answered *yes*. Your installation EXEC uses the second argument (the application identification code) to install additional features (if there are any) of your application.
3. The INSTFPP EXEC has access to the following minidisks:
   - The equivalent of 10M of DASD (for example, approximately 20 cylinders of IBM 3380 DASD), accessed as a temporary disk at virtual address 002 using a mode of **A**. Use this minidisk as your work space and for assemblies.
   - The equivalent of 5M of DASD (for example, approximately 10 cylinders of IBM 3380 DASD), accessed as a temporary disk at virtual address 001 using a mode of **C**. This minidisk is used in the installation process to load the first two logical files from the application tape.
   - MAINT's 191 minidisk, accessed as **D**
   - MAINT's 193 minidisk, accessed as **T**
   - MAINT's 319 minidisk, accessed as **P**.
4. The customer has defined a specific minidisk for your application or has enough DASD space for your application on a minidisk that contains other application files.
5. Log off user IDs that own minidisks onto which you will be loading files.

   For example, your application may load files to a virtual machine that usually runs disconnected. This virtual machine must be logged off for MAINT to be able to use its minidisks during the installation.
6. Link to proper minidisks in write mode.

   Linking to minidisks should be done using virtual link addresses ranging from 210 to 280. Passwords, if provided in the preinstallation documentation samples, may be hardcoded in the EXEC. If the link fails, prompt the customer for the password.

   Remember that, if you use REXX to code your EXEC, you cannot code the minidisk password on the same line as the link command unless you precede the link command with the ADDRESS COMMAND statement.
7. Access necessary minidisks using filemodes **E** through **O.**
8. Format any application-specific minidisks.

   Note: Do not format any common minidisks (for example, MAINT's 19E or 319 minidisk) if you intend to install your application on that minidisk. Formatting a common minidisk destroys all files on that minidisk.
9. Save spool state parameters for spooling unit record devices back to their initial parameters before exiting.

   For example, if part of the verification of an application "punches" a file to another user, the spooling parameters for the punch should be saved and then returned to their original values after the test spooling is complete.
10. The tape is now positioned at the beginning of the third logical tape file.

11. Load the application files from the tape to the appropriate minidisks and keep track of the tape position as the files are loaded. The EXEC should perform error checking after each load and, if an error occurs, should advance the tape to the end of the application.

12. Generate the saved segment, if there is one.

    If your application generates a saved segment during installation, verify that storage is large enough before attempting the generation. If storage is too small, send a message telling the customer how to correct the situation and begin the generation procedure again.

    The installation EXEC calls the SNTINFO module to obtain information about a DMKSNT entry. The SNTINFO module provides the saved segment's start address, the end address, the size, and the condition code. For more information about the SNTINFO module, see the *VM/SP Installation Guide* or enter:

    ```
    help sntinfo
    ```

    on VM/IS to see the online help information.

13. Send messages to the customer telling the status of the installation and verification process.

    Use the following messages, or combinations of messages, as they apply to your application installation.

    - For all EXECs:

```
*** <prodid> installation in progress...

*** <prodid> loading files to <userid> <vaddr> disk...
```

    - For EXECs using the 0 (zero) return code:

```
*** <prodid> installation successful.

*** <prodid> verification in progress...

*** <prodid> verification successful.
```

      Note: The return code is 0 (zero) when your application is installed, and the verification is successful.

    - For EXECs using the 777 return code:

```
*********************************************************
*** <prodid> - <product name>
*** All necessary files have been loaded successfully.
*** Refer to Innnnnnn MEMO to complete installation.
*********************************************************
```

      Note: The return code is 777 when your application files are loaded from tape, but additional application-related installation tasks are required to complete the installation and do the verification. For example, a disconnected virtual machine may need to be logged on, storage may need to be redefined, or some individual application tailoring may be required.

- For EXECS using the 888 return code:

```
*** <prodid> installation successful.

*** <prodid> must be verified manually.
```

Note: The return code is 888 when your application is installed, and the verification does not run. Verification must be done manually.

<prodid> is either an application identification number or application name, <userid> is the user ID, <vaddr> is the virtual address of the minidisk to which application files are being loaded, and <product name> is the name of the application.

To make sure messages do not go unnoticed, clear the screen before issuing the message, or present a prompt that says:

```
Press ENTER to continue
```

Use caution when displaying messages by clearing the screen. Remember that important information may be lost if the screen clears to display a message.

Note: The above listing is not a complete list of informational and error messages. Use other messages where appropriate, depending on your requirements.

14. Before exiting:
    - Release and detach all application related minidisks and temporary minidisks. Do not detach any minidisks that are not application specific and are shared by other products (for example, MAINT's 19E minidisk).
    - Spool devices back to their original parameters.
    - Erase from MAINT's minidisks any files that may have been generated by the installation process and are no longer needed.
15. Perform error checking for any CMS and CP commands your Installation EXEC issues.
16. If an error occurs, position the tape at the end of the last application file being installed and exit with the appropriate error return code and message.
17. When the installation is complete, position the tape at the end of the last application file before exiting.
18. Set the value of the return code into a variable, and exit with the variable.
19. Include a prologue or an epilogue to explain the EXEC.

Notes:

1. Your application installation EXEC must exit with the appropriate return code as explained below:
    a. The return code is 111 when the customer decides to exit from the installation procedure.
    b. The return code is 999 when your application is installed, but the verification fails.
    c. Any return code other than 0, 111, 777, 888, or 999 indicates a failure in the installation process and is recorded as such in the PROD LEVEL file. In the Memo to Users file, you must document any other return codes that are set up in your application installation EXEC.

2. Your application installation EXEC **must** keep track of where it is in the installation process and position your application tape to the end of the last file of your application before exiting from the EXEC. Your EXEC should never rewind the tape because there may be other applications or features following your application.

The Verification Routine

You create a verification EXEC and supply it in logical tape 5. Your application installation EXEC should load your verification EXEC to one of the application's production minidisks where other executable code is loaded.

Note: When creating your verification EXEC, use the file naming convention specified in "The Package Tape Format" on page 64.

It is recommended that you use REXX (the System Product Interpreter Language) to write your verification EXEC.

Whether or not you supply a verification EXEC, document manual verification procedures in the Memo to Users file for your application.

If you do supply a verification EXEC, you can automatically invoke this EXEC (from the Installation EXEC) on the MAINT user ID. Or instead, you could have the customer log on to the CMSUSER user ID to verify successful installation. Verification should always be performed before any tailoring is done.

The CMSUSER user ID is a general class user ID, of storage size 2M, that can verify whether your application is installed and running properly for general class user IDs. Applications that have their own user IDs may verify that their application are installed and running properly for general class users from one of the application's user IDs instead of from the CMSUSER user ID.

If your application includes more than one feature, each feature should be packaged so it can be verified independently. Tools that you supply to assist the customer in tailoring the application do not have to be verified.

Your application verification EXEC should do the following:

1. Check for the existence of any prerequisite program or EXEC before attempting the verification process. Invoke only those applications indicated as prerequisites, and provide error messages if the required applications are not present.
2. Save spool state parameters and return devices to their initial spool states, if the EXEC changes those parameters.
3. Link to appropriate minidisks.

   When linking to minidisks, use virtual link addresses ranging from 210 to 280. Passwords, if provided in the preinstallation documentation samples, may be hardcoded in the EXEC. If the link fails, prompt the customer for the password. Remember that if you use REXX to code your EXEC, you cannot code the minidisk password on the same line as the link command unless you precede the link command with the ADDRESS COMMAND statement.
4. Never format any minidisks that could be shared by other products (for example, MAINT's 19E minidisk).

5. Define and format a large enough temporary minidisk if temporary files are generated or if disk space is required by the verification process.
6. Access the minidisks using filemodes **E** through **O**
7. Execute one or more product functions to verify that your application runs properly. Document expected results in the Memo to Users.
8. Erase unnecessary files from MAINT or CMSUSER minidisks that may have been generated by the verification process.
9. Release and detach all application specific minidisks and any temporary minidisks before ending. Minidisks that are not application specific (for example, MAINT's 19E minidisk) should not be detached.
10. Perform error checking for any CMS and CP commands your verification EXEC issues.
11. Issue a message to the customer stating whether or not the verification completed successfully and issue a return code.
12. Include a prologue or an epilogue to explain the EXEC.

The Memo to Users

The Memo to Users provides information to help the customer install and verify your application. The Memo to Users is a print file that must be generated with carriage controls in column 1 or with column 1 left blank. The memo is printed by the INSTFPP EXEC by the following command:

```
PRINT Innnnnnn MEMO C (CC
```

where *nnnnnnn* is your application number. For information on the suggested contents of the Memo to Users and the task of writing it, see "The Memo to Users" on page 87.

Building the Package Tape

Build your tape using the VMFPLC2 command. You use this command to dump files from disk onto tape, and to control a specific tape drive. Files processed by the VMFPLC2 command must be from CMS-formatted minidisks.

See the *VM/SP Installation Guide* for more information on the VMFPLC2 command.

You can write an EXEC that uses the VMFPLC2 command to create an application tape. See Figure 12 for an example of such an EXEC.

Note: The example EXEC using the VMFPLC2 command calls other EXECs created specifically to facilitate the tape build operation. See "Using the LISTFILE Command" on page 77 for information on creating these EXECs.

```
/*********************************************************************
*
* Application 3927A01 Install Tape Creation EXEC
*
* TAPE FORMAT:
* 1- I3927A01 012005       Application Identifier File
*    I3927A01 EXEC         Application Installation EXEC
*    TM                    Tape mark
* 2- I3927A01 MEMO         Memo to Users File
*    TM                    Tape mark
* 3- SOURCE                Source Code Files
*                          (non-executable code)
*    TM                    Tape mark
* 4- TUTORIAL              Online Documentation
*                          and Courses
*    TM                    Tape mark
* 5- APPLICATION FILES     Your Application Files
*    TM                    Tape mark
* 6- TM                    Tape mark
*********************************************************************/
ADDRESS COMMAND
'CP REWIND 181'                  /* Rewind the tape to position it  */
'VMFPLC2 MODESET (DEN 1600'      /* Set the tape density            */
'VMFPLC2 DUMP I3927A01 012005 C' /* Dump Application Identifier and */
'VMFPLC2 DUMP I3927A01 EXEC   C' /* Application Installation EXEC   */
'VMFPLC2 WTM'                    /* Write a tape mark               */
'VMFPLC2 DUMP I3927A01 MEMO   C' /* Dump Memo to Users File         */
'VMFPLC2 WTM'                    /* Write a tape mark               */
'EXEC S3927A01 VMFPLC2 DUMP'     /* Dump Source Files               */
'VMFPLC2 WTM'                    /* Write a tape mark               */
'EXEC T3927A01 VMFPLC2 DUMP'     /* Dump online information panels  */
'VMFPLC2 WTM'                    /* Write a tape mark               */
'EXEC A3927A01 VMFPLC2 DUMP'     /* Dump your application files     */
'VMFPLC2 WTM 2'                  /* Write two tape marks            */
'CP REWIND 181'                  /* Rewind the tape                 */
EXIT
```

Figure 12. Example Tape Build using VMFPLC2

Note that the EXEC does the following:

- Rewinds the tape to position it correctly
- Sets the tape density to 1600 bits per inch
- Dumps the application identifier file and the application installation EXEC
- Writes a tape mark to indicate the end of logical file 1
- Dumps the Memo to Users file and writes a tape mark
- Dumps the source files and writes a tape mark
- Dumps the online information files and writes a tape mark
- Dumps the application files and writes two tape marks
- Rewinds the tape.

Using the LISTFILE Command

The source files, online information files, and application files are dumped to tape using EXECs called S3927A01, T3927A01, and A3927A01 respectively. Each of these EXECs contains a list of files to be dumped. You can create such an exec using the LISTFILE command.

For example, if all of the source files use a filetype of 3927SRCE, you could create the S3927A01 EXEC file by entering the following commands:

```
listfile * 3927srce * (exec
rename cms exec a s3927a01 = =
```

The first command creates a file containing a list of all files with a filetype of 3927SRCE and names the file CMS EXEC. The second command renames the file to a name that is recognized and used by the application tape creation EXEC. (See Figure 12.)

For more information on using the VMFPLC2 command, see the *VM/SP Installation Guide*.

Note: Remember that the CMS EXEC you create using the LISTFILE command includes a filemode for each file listed. If you reaccess your minidisks using different CMS mode letters, your EXEC may not work.

System Restrictions

Some minidisk addresses and filemodes are not usable by your EXECs because VM uses them for other purposes. Following are minidisk addresses and filemodes that are available for use in your application installation and verification EXECs.

- During installation, filemodes **E** through **O** are available as access modes.
- Several MAINT-owned virtual minidisk addresses are not available for loading application files. To obtain a list of minidisks that are used by VM/IS, log onto the MAINT user ID and issue the command:

  ```
  DIRMAINT REVIEW
  ```

- A MAINT minidisk address is required during installation to link to each minidisk owned by any other user ID. Addresses available for linking to another user ID's minidisks are 210 through 280. MAINT, in most cases, does not have any of its own minidisks defined and linked within this address range.
- User IDs should relate to the name of your application. User IDs already included in the system samples, that you cannot use, are documented in the USER DIRECT file located on the ***DIRMAINT*** 195 minidisk. To get a copy of the USER DIRECT file, log on to the ADMIN user ID and use the available VM/IPF facilities to access the USER DIRECT file. More information on obtaining a copy of the USER DIRECT file is available in the *VM/IPF Administration* manual.

Testing for Successful Installation

After you have completed building the package tape, you should test your installation procedures. Have a system administrator install your application to make sure that your installation procedure is clear and correct.

For information on testing your installation procedure and your package, see Chapter 6, "Testing Your VM/Integrated System Application" on page 57.

Summary

To successfully package a VM/IS application, you should do the following:

- Learn and understand the suggested format and restrictions for building the package tape.
- Understand the VM/SP installation procedure and the INSTFPP EXEC.
- Understand the requirements for and create an installation EXEC. Your application installation EXEC should do the following:
  - Use filemodes **E** through **O**.
  - The required entries in the VM/IS BASE were added as instructed in the application documentation (for example, directory entries or DMKRIO entries).
  - The customer defined a specific minidisk for your application, or made sure sufficient DASD space exists to load your application to a minidisk that contains other application files.
  - Format any application-specific minidisks.
  - Be executed by INSTFPP from the MAINT user ID, which defines virtual storage to be from 12M to 16M, and initiates CMS.
  - The tape is positioned at the beginning of the third logical tape file.
  - Keep track of tape position as the files are loaded.
  - Log off user IDs that own minidisks onto which you will load files.
  - Save spool state parameters for spooling unit record devices back to their initial parameters.
  - Use a temporary disk when assembling or compiling programs.
  - Link to proper minidisks in write mode and then load files from the tape.
  - Generate the shared segment, if there is one.
  - Send messages to the customer telling the status of the installation and verification process.
  - Release and detach all application-related minidisks before exiting.
  - Spool devices back to their original parameters before each exit from the EXEC.
  - Set the value of the return code into a variable, and exit with the variable.
  - Upon error, position the tape to the end of the application being installed, and exit with the appropriate error return code and message.
  - Include an epilogue that explains the EXEC.
- Understand the requirements for and create a verification routine. Your verification routine should do the following:
  - Check for the existence of any prerequisite program or EXEC before attempting the verification process.
  - Save spool state parameters for spooling unit record devices back to their initial parameters, if the EXEC changes those parameters.
  - Link to appropriate minidisks used in verifying the installation and functioning of your application.
  - Execute one or more applications to verify that your application runs properly. Document expected results in the Memo to Users.

  - Release and detach all minidisks linked before exiting.
  - Spool devices back to the way they were before each exit.
  - Issue a message to the customer stating whether or not the verification was completed successfully.
  - Include an epilogue that explains the EXEC.
- Write the Memo to Users and include it on the tape.
- Build the package tape using the VMFPLC2 command with the appropriate parameters.
- Test that your package can be successfully installed from the tape.

Chapter 8. Documenting a VM/Integrated System Application

Your application tends to be useful only if your customers are able to understand and use it. In most cases, you must document the application.

Good documentation is an important part of a finished product. You should develop documentation that concisely describes the capabilities and operation of your application.

This chapter guides you in writing complete and appropriate documentation to accompany your application.

Introduction to Documenting

Documenting can mean different things. For example, your documentation could take the form of a manual, or it could be online help panels. In either case, you are presenting the customer with information about your application and how to use it.

You may be able to put all necessary information in the form of help panels, eliminating the need for a hard-copy manual and reducing costs. Some information may have to appear as hard copy. For example, installation information and source listings might be better in a printed format.

You should decide on the type of documentation you will use during the planning phase of your project, and develop it as an integral part of your application.

The Publications Library

Designing your library involves two areas. You have to decide on the physical appearance of the library as well as on the content.

You should make your library similar in appearance[4] to the existing VM/IS library so that they will blend into the customer's library. Your manuals will then look as if they were designed for the VM/IS system.

You may have reasons to use a different format. For example, you may have illustrations or tables that cannot be easily produced using the VM/IS library format. You should investigate your chosen format during your planning phase and ensure that you have chosen an acceptable and usable format.

In designing your library's content, you could choose from several typical types of manual, either singly or in combination:

- General Information
- End-user
  - Primer
  - Reference
  - Messages.

[4] When designing your covers, don't forget that IBM is a registered trademark. You must replace any occurrence of IBM on your manual covers with the name of your own company.

- Administration
  - Primer
  - Installation
  - Customization
  - Memo to Users
  - Reference
  - Messages.
- Operation
  - Operating
  - Reference
  - Messages.
- System programmer.
  - Program directory
  - Installation
  - Memo to Users
  - Messages
  - Program Description
  - Operations.

Remember that VM/IS assumes that a customer may not have a system programmer on his staff. You should design your publications for the end-user and the administrator. In most cases, the only documentation you need to supply is:

- End-user primer
- Administration primer
- Memo to Users (supplied as a print file only).

Note: We suggest you name your end-user primer *Working With (fill in the name of your product)* and that you name your administration primer *Managing (fill in the name of your product)*. These titles match the titles used for existing VM/IS primer-style manuals and blend your books into the existing library.

This chapter discusses only these three pieces of documentation. These manuals may not be the only documentation that you produce. Your own case may dictate that you produce a more extensive library to adequately document your application.

To blend into the existing VM/IS library, however, we recommend that you produce at least the three above-listed pieces of documentation.

Designing the Publications to VM/Integrated System Standards

To make your publications blend in with the VM/IS library, use the same format for the manuals and, if possible, the same style of writing.

Plan your manuals to look like those in the VM/IS library. Use the same page size, the same cover design, and the same layout. In addition, plan to produce your manuals using a task-oriented format. That is, describe the tasks that you expect the user to perform and the steps the user follows to perform the tasks. Try to avoid providing only a reference-style manual that describes your application's functions.

You may also have to produce a reference manual. If your application is large, you could have difficulty describing in a primer-style manual all the tasks the user might perform. In this case, consider producing a reference manual for those users who venture beyond the limits of the primer. Another thought is to produce a primer as

two manuals with a part one to cover the most important tasks and a part two to cover the less important or less needed tasks.

Another consideration when producing documentation is the possibility that the document might be translated into other languages to make your application marketable in other countries. This consideration has a bearing on all information you produce, including the online tutorial, the help panels, your application menus, online messages, and your documentation. Considerations include:

- Avoid using words or combinations of words that might be difficult to translate or misleading. For instance, you may use an instance that uses the character string *yourname* to indicate where you want the end-user to insert his own name. Because this is a combination of two words, a translator may not understand, or may translate the string into something totally different.
- Remember that languages other than English often use more words to express the same idea. In designing panels, try to leave about thirty percent of the panel unused for translation purposes. On your panels, use field descriptions that can be changed easily without redesigning the panel.
- The same guidelines apply to messages. Because of their nature, online messages tend to be cryptic. They can be difficult to understand and more difficult to translate. In addition, remember that there are often size limitations on messages, imposed by the method used to display them. For example, instead of clearing the screen to display a message, a menu could allocate one or two lines to display messages. If your message uses all the space available, the message might not be readily translatable.

There are other considerations to keep in mind when developing information for translation. Try talking to the person doing the translating if this is possible.

Writing the Publications

This section discusses the information you should supply for your application. Information covered in this section includes the end-user primer, the administration primer, and the Memo to Users, which is shipped on the package tape.

Remember that the primer-style manuals for the end-user and for the administrator should not necessarily be designed to cover everything. Rather, you should design the manuals to cover the most important aspects of your application. If your application contains functions that you would not cover in the context of primer-style manuals, you should consider producing end-user and administrative reference manuals.

End-user Documentation

You should pattern your end-user primer after the existing VM/IS primers. You should produce a task-oriented manual that includes a large number of illustrations and examples, and uses color to highlight user input. Wherever possible, use numbered step-by-step instructions to describe the tasks.

Your manual should include some or all of the following:

- Introduction
- Most important tasks
- Examples
- Glossary.

These subjects are discussed in more detail in the following sections.

Introduction

The first section of your manual should introduce the end-user to your application. Present an overview of your application and the types of tasks for which the application is designed.

Most Important Tasks

Before beginning this section of the end-user primer, you should perform an analysis of the application to determine the tasks that are performed by a user. Further, you should determine which of these tasks can be considered to be major tasks and which are subtasks. You should then arrange the tasks in order of probable frequency of use.

Develop the steps the user would normally take to perform the task and present them in a step-by-step format. Each step represents one user action and system reaction. Try to illustrate any relevant information that occurs during the performance of a step. For example, if the system or the application produces any messages as the result of a step, reproduce the messages so the user will know what to expect.

Examples

Use examples extensively throughout your manual. Often an example illustrates the required input or the expected output more concisely than a large amount of text.

Glossary

If your manual uses acronyms or terms that apply specifically to your application, you should consider including a glossary. In addition, highlight the first occurrence in the text of any term that appears in the glossary.

Administration Documentation

The system administrator, in most cases, is not a system programmer. In addition, the administrator may only perform the administration function on a part-time basis. He or she may also act as the system operator, as well as being an end-user.

You should pattern your administration primer after the existing VM/IS administration primers. You thereby produce a task-oriented manual that includes a large number of illustrations and examples, and uses color to highlight user input. Wherever possible, use numbered step-by-step instructions to describe the tasks.

Your manual should include some or all of the following:

- Introduction
- Installing and customizing
- Operating and administering
- Servicing
- Migrating
- Contents of certain files
- Glossary.

These subjects are discussed in more detail in the following sections.

Introduction

The first part of your manual should give the administrator an introduction to your application and an overview of the administration tasks that your application requires. You should not have to introduce the administrator to VM/IS, because other manuals do so.

If your application depends on other products, either as a prerequisite or as a corequisite, you should point that out in this section.

Installing and Customizing

This information should already appear in the Memo to Users that you include as the second logical file on your package tape. (For information on the content of the package tape, see Chapter 7, "Packaging a VM/Integrated System Application" on page 63. For information on the Memo to Users, see "The Memo to Users" on page 87.) If this is the case, you do not need to duplicate the information in your administration manual.

You may want to provide an overview of each process and point to the Memo to Users for more information. You can also include additional information as a supplement to the information contained in the Memo to Users.

Operating and Administering

You can cover these subjects in either one section or two, depending on the quantity of information that you must supply.

The operating information should include any startup and shutdown information. If your application uses one or more service machines, the user IDs may be autologged each time the operator performs a system IPL. If the service machines are not autologged, include a step-by-step procedure to bring up your application.

Document any shutdown requirements. If the application shuts down prematurely, do users lose data in files that are not properly closed? Are there functions that must be shut down in a specific sequence?

Your administration information should include any information the administrator needs to keep the application running efficiently.

If new users must be enrolled to use the application, provide a step-by-step procedure for enrollment. Include such information as minidisk links that the administrator must add to the new user ID. The user may need copies of files on his or her personal minidisk to ensure proper operation of the application. Outline the procedure for copying the files (or include an EXEC to perform the procedure), and indicate what customization of these files, if any, must be done.

Your application may produce print files, punch files, or reader files. If these files are needed only part of the time, for example, for problem diagnosis, provide a procedure to purge or archive the unneeded files to conserve spool space. If possible, try to make the procedure automatic.

You may want to include a section on problem handling and recovering from application or system errors. If your application includes error or information messages, you may want to include explanations of the messages.

Note: If your application includes a large number of messages, you might want to provide an appendix or even a separate manual for the messages.

Administrative procedures should appear in a step-by-step format whenever possible. Try to supply sufficient information for the administrator to do his or her job without going into fine detail. Remember that a lot of explanatory information could confuse an administrator who is not trained as a system programmer.

Servicing

You decided on your service strategy in your planning and designing stages. In this section, outline to the customer the support you intend to supply for your application.

You may already have discussed the service strategy in the Memo to Users. If so, it should not be necessary to duplicate the information in your administration manual. Rather, you should provide supplementary information.

You should indicate who the customer contacts for service. Outline the format in which you intend to apply service and the skills needed to apply the service. In addition, point out any files that you use to record the history of your application (for example, a SERVICE LEVEL file) and how to interpret the information contained in such files.

If you have decided to supply service only in the form of a complete update of the application, you should document how often you plan to update your application.

Migrating

An important part of VM/IS is its ability to migrate to new releases as they become available. Because of this, you must document information about the migration process. Remember that you are not telling the customer how to migrate but rather are outlining the migration process so the customer can plan.

You may want to include recommendations for the customer to facilitate the future migration process. For example, if the customer has a choice of DASD volumes on which he can store your application files, you might want to recommend the volume he or she should (or should not) use for storage.

Contents of Files

You may want to list the contents of some of the files included in your application. Some files that you might want to list are:

- Files that the customer can modify or customize
- Examples of files the customer must create
- Samples of directory entries and PROFILE EXEC files that might be used by general users and your service machines to illustrate minidisk links required by your application
- Specifications for the DMKSNT file
- If your application uses a saved segment, a sample DMKSNT *NAMESYS* macro showing details of areas that may change.

Consider listing a specific file when the customer must customize the file, and the contents of that file are critical to the operation of the application. You might even decide to highlight the sections that must be customized.

In addition, when the customer customizes a critical file, there is always the danger of he or she making a mistake that can affect the operation of the application. If this happens, the customer must reverse the change to continue using the application; when customers have already customized other parts of the file, they may not want to lose their other changes by restoring the original file from the package tape.

They might find it easier to change the file by comparing it to a printed listing of the original version.

Sometimes a customer may need to create files for the application. These files may be specific to the location, or they may be specific to each user ID that uses the application. In either case, the customer would probably find an example of the file helpful.

If you supply a tool to simplify customizing files (using one or a combination of XEDIT, REXX, or ISPF), be sure to include instructions for the use of the tool.

If your application uses a service virtual machine, other user IDs may need to link to the service machine's minidisks to access the application. In addition, the administrator may have to create the service machine as a part of the installation. A sample directory entry for the service machine assists the administrator in creating the service machine. A sample user's directory entry ensures that the administrator is providing the proper links for other users. Or, you may decide to make the minidisk links through the user's PROFILE EXEC. In this case, you might want to supply a sample file or even sample lines to add to existing files. In either case, a sample usually helps and is appreciated.

Glossary

If you use a large number of acronyms or terms that have special meaning for your application, consider including a glossary of terms. You should highlight the first occurrence of each term in the text of your manual to indicate to the reader that the term appears in the glossary.

The Memo to Users

The customer may not be a system programmer. The Memo to Users contains helpful information to the customer and is a quick means of installing your application and verifying the successful installation.

You supply the Memo to Users as the second logical file on your package tape. Name the file Ixxxxxxx MEMOii, where *xxxxxxx* represents your application number and *ii* represents the alphanumeric identification code that identifies individual features of your application. The file is a print file and must be generated with carriage controls in column 1 or with column 1 left blank. The memo is printed by the automatic installation process.

Your Memo to Users should contain information on the following topics and in the following order:

1. Preinstallation
2. Prerequisites
3. Corequisites
4. Installation
5. Verification
6. Tailoring
7. Error and Information Messages
8. Service
9. Special Instructions.

Note: Remember that your Memo to Users should also contain a Table of Contents to help the customer find individual sections.

The following sections describe the included topics.

Preinstallation Information

The hard copy documentation you package with your application tape should explain any system additions required to prepare the system for loading and generating your application. This preinstallation information can also be documented in the Memo to Users you ship on your application tape. Remember that the customer would need to mount the tape and use the INSTFPP EXEC to print your application memo before being able to read the information. You should provide this information as part of the hard-copy documentation you supply with your application package.

Your preinstallation documentation should:

- Explain how to invoke the INSTFPP EXEC, which in turn, invokes your application installation EXEC.

  For details on the INSTFPP command, see the *VM/SP Installation Guide*, or enter:

  ```
  help instfpp
  ```

  on VM/IS to see the online help information for the command.

- Specify space requirements.

  Identify the size, in 1K, 2K, or 4K blocks, required to load your application files to an application minidisk containing other applications, or for each minidisk if your application is installed on its own minidisks.

- Supply the following directory information in sample form, when an application has its own disconnected virtual machines, or its own user IDs:

  - User ID name
  - Virtual storage size
  - Maximum virtual storage size
  - User privilege classes
  - Priority used by CP priority dispatcher
  - Minidisk addresses
  - Minidisk sizes, in blocks and cylinders, required to load the application files to a common application minidisk
  - Block size used for size requirement above
  - Any other directory information; for example, how to link to other minidisks, obtain account information, or load CMS.

  Figure 13 shows an example directory entry for the SQLDBA user ID.

- Point out the minimum virtual storage required to run your application.

- DMKRIO entries: If your application requires special hardware (real I/O devices), supply the necessary specific information for the DMKRIO file.

- DMKSNT entries: If your application requires a saved segment, supply these entries.

- Miscellaneous: Include any other preinstallation actions required for your application.

Instruct your customer to mount your application tape and use the INSTFPP EXEC after the system is set up. Document the expected outcome, and refer your customer to the Memo to Users printed during the installation for post-installation information.

```
USER SQLDBA xxxxxxxx 6M 6M G
 ACCOUNT SQLDBA xxxxxxxx
 OPTION REALTIMER MAXCONN 25
 IUCV ALLOW
 IPL CMS
 CONSOLE 009 3215 T OPERATOR
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1443
 LINK MAINT 190 190 RR
 LINK MAINT 19D 19D RR
 LINK MAINT 19E 19E RR
 MDISK 191 3380 595    009  PROFPK  W  rrrrrrrr wwwwwwww
 MDISK 193 3380 604    030  PROFPK  R  rrrrrrrr wwwwwwww
 MDISK 195 3380 634    010  PROFPK  RR rrrrrrrr wwwwwwww mmmmmmmm
 MDISK 200 3380 644    024  PROFPK  R  rrrrrrrr wwwwwwww
 MDISK 201 3380 668    010  PROFPK  R  rrrrrrrr wwwwwwww
 MDISK 202 3380 678    080  PROFPK  R  rrrrrrrr wwwwwwww
```

Figure 13. Example Directory Entry For SQLDBA

A description of the directory entries follows:

`USER SQLDBA xxxxxxxx 6M 6M G`
: This line contains:
  - User ID (SQLDBA)
  - Password (xxxxxxxx)
  - Logon virtual storage size (6M)
  - Maximum virtual storage size (6M)
  - Privilege class (G).

`ACCOUNT SQLDBA xxxxxxxx`
: This line is for accounting recording.

`OPTION REALTIMER MAXCONN 25`
: This line updates the virtual machine's timer during wait state, and sets the maximum number of allowable Inter-User Communication Vehicle (IUCV) connections to 25.

`IUCV ALLOW`
: This line allows any other virtual machine to establish a communication path with this virtual machine.

`IPL CMS`
: This line loads CMS when the user ID logs on.

`CONSOLE 009 3215 T OPERATOR`
: This line sets up the user's console at address 009 as a type 3215, and sets the Operator user ID as the secondary console.

`SPOOL 00C 2540 READER *`
: This line sets up the user's reader (00C) as type 2540 spooled to class ANY.

`SPOOL 00D 2540 PUNCH A`
: This line sets up the user's punch (00D) class A.

`SPOOL 00E 1403 A`
: This line sets up the user's printer (00E) class A.

`LINK MAINT 190 190 RR`
: This line allows the SQLDBA user ID to link to MAINT's 190 minidisk at SQLDBA's virtual address 190 with read only (RR) access.

```
MDISK 195 3380 634 010 PROFPK RR rr... ww... mm...
```

This line describes one of the user's own minidisks with this information:

- Minidisk address (195)
- DASD type (3380)
- Starting cylinder location (634)
- Number of cylinders (010)
- Real DASD volume label (PROFPK)
- Read only access (RR)
- Read password (rrrrrrrr)
- Write password (wwwwwwww)
- Multiwrite password (mmmmmmmm).

Note: The information in this line has been compacted for illustration purposes. In the actual directory entry, the line items would be spaced farther apart, and the read, write, and multiwrite passwords could be up to eight characters long.

Generally, the system administrator does not perform direct updates to the CP directory. Usually, the system administrator uses VM/IPF. Supply a script of an VM/IPF session to help the system administrator update the CP directory. Refer to *VM/IS Managing Your System* for more information on the VM/IPF script.

For more information on directory entries and disconnected virtual machines, see the *VM/SP Planning Guide and Reference*.

Prerequisites

List any hardware or software required before your application can be installed and made operational.

Corequisites

List any hardware or software required to support some of the functions of your application.

Installation

This section contains the following:

- Information on the location of the application files after your application installation is complete.

  To illustrate, the accounting package previously used in an example may load the execution files to the MAINT 31A minidisk. Files for regenerating or servicing the package are not loaded to DASD, but may be found on the third logical tape file of the package's application tape.

- Build or generation procedures.

  These procedures might include how to create and save the saved segment for your application or how to initialize a service machine.

- Instructions for modifying any ***load-and-go*** parameters.

  The load-and-go version may have been set up with certain defaults. If the customer wishes to modify these defaults, and certain files are required, the instructions should clearly indicate how to modify these defaults.

Verification

This section contains the following:

- Instructions for testing and verifying correct installation of your application using the verification EXEC. These instructions should indicate the expected results.

  The suggested form of these instructions is a copy of the terminal interactions as contained in a console file.

- The permanent location of the verification EXEC.
- A manual verification procedure. You should include this procedure even if your application is automatically verified.

Tailoring

This section contains the following:

- Manual tailoring instructions that may be required to complete application installation or generation.

  If your application includes a service virtual machine, and you shipped the machine's directory entry with the password set to *NOLOG*, include instructions for changing the password.

- Optional tailoring to customize your application.

  If your application includes a disconnected virtual machine, the customer could ensure that this machine is ***autologged*** after system IPL by editing the PROFILE EXEC belonging to the AUTOLOG1 user ID.

- A reference to your application documentation for extensive tailoring.

  If extensive tailoring is required, it need not be completely documented in the memo, but the memo should also refer the customer to your documentation to find these instructions. Include information about any tools you supply to simplify the tailoring process.

- A reference to the PFSETUP EXEC.

  If necessary, you should inform the customer to run PFSETUP to reflect the new tailoring for the VM/IS menus.

Error and Information Messages

This section contains the following:

- Return codes other than 0, 111, 777, 888, or 999 caused by your application installation EXEC, and explanations for each. If the messages can be found in an application manual, the memo should refer to that manual by name and number.
- Error return codes for your verification EXEC.
- Special instructions, hints, cautions, samples, or examples.

  These would include information on appropriate actions to try and things to check if the installation or verification does not work.

Service

This section contains the following:

- A description of the way the service is supplied for your application .
- Any preservice procedures that may be required or are recommended.

Special Instructions

This section contains the following:

- Password information. If a customer changes passwords in the directory after installation of your application, you must identify all EXECs and files that also require the new passwords. You must also provide instructions on changing passwords.
- Any additional information the customer should know about your application.

The Documentation Tools of VM/Integrated System

VM/IS provides you with several tools to help you produce your documentation. These tools fall into the following categories:

- Editors
- Formatters
- Graphics support
- File management
- Personal Computer support
- Printer support
- Administrative
- Batch facilities
- Communications.

Using the tools supplied with VM/IS, you can develop your formal documentation, then produce it in a format that can be used by a publisher to print your manuals. The following sections discuss the tools further.

Note: This chapter assumes that your location has installed the VM/IS BASE and all optional packages. If you do not have all optional packages installed, you may not have access to some of the tools listed in this section.

Editors

VM/IS supplies the following products to assist you in creating and editing your documentation files:

- Application System
- ***DisplayWrite/370***
- ISPF/PDF
- System Product Editor (XEDIT).

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Formatters

VM/IS supplies the following products to assist you in formatting your documentation files:

- DCF
- DisplayWrite/370.

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Graphics Support

VM/IS supplies the following products to assist you with graphics support for your documentation:

- GDDM
- GDDM-PGF.

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

File Management

VM/IS supplies the following products to assist you in managing your data files:

- ISPF/PDF
- Application System
- ***CICS/VM***
- ***DBEDIT***
- ***VM FSF***
- ***VSE/VSAM.***

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Personal Computer Support

VM/IS supplies the following products to assist you in using PC products for your documentation files:

- ***3270 PC File Transfer***
- ***ECF***
- PROFS Application Support Feature.

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Printer Support

VM/IS supplies the following products to assist you in printing your documentation files:

- PROFS Print Services Facility
- System printer support (included in VM/SP)
- VM3812.

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Administrative

VM/IS supplies the following products to assist you in administering your application project:

- Application System
- PROFS
- PROFS Application Support Feature.

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Batch Facilities

VM/IS supplies the following product to assist you with time consuming batch jobs:

- VM Batch Facility.

For information about this product, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Communications

VM/IS supplies the following products to assist you in communicating with other users or systems:

- ACF/VTAM
- *CVIEW*
- PROFS
- PVM
- RSCS
- TCP/IP.

For information about these products, see Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 and the product documentation.

Summary

Document your application using the following suggestions:

- Use the planning phase of your project to determine the type of documentation you should use.
- Make your library similar in appearance and format to the existing VM/IS library.
- Although you can choose from several types of manual, you should supply at least an end-user primer and an administration primer. In addition, you should always supply a Memo to Users as a print file on the package tape.
- Your end-user manual should include some or all of the following topics or sections:
  - Introduction
  - Most important tasks
  - Examples
  - Glossary.

- Your administration manual should include some or all of the following topics or sections:
  - Introduction
  - Planning, installing, and customizing
  - Operating and administering
  - Servicing
  - Migrating
  - Contents of certain files
  - Glossary.
- Your Memo to Users should include some or all of the following topics or sections:
  - Preinstallation
  - Installation
  - Verification
  - Tailoring
  - Error Messages
  - Prerequisites
  - Corequisites
  - Service
  - Special Instructions.
- Plan to use tools supplied by VM/IS in the following categories to help you document your application:
  - Editors
  - Formatters
  - Graphics support
  - File management
  - Personal Computer support
  - Printer support
  - Administrative
  - Batch facilities
  - Communications.

Part III. VM/Integrated System Packages, Functions, and Products

Chapter 9. The VM/Integrated System Products and What They Do

This chapter lists the VM/IS packages and the products included in each package. For a short functional description of each product, see Chapter 10, "What the VM/Integrated System Products Do" on page 107.

The VM/Integrated System Packages

Although VM/IS is composed of several products, not all are required for application programming purposes and thus, not all have associated application development information. Please note that this publication discusses only those products that provide application development information in their publications.

The following tables list, by package, all products offered through VM/IS.

The Base

VM/IS offers a base known as VM/IS BASE. VM/IS BASE consists of a number of functions equivalent to the products shown in Table 3 (the Core Base) and in Table 4 on page 100 (the Additional Base).

VM/IS Function Name	Equivalent Product Name	Equivalent Version/ Release/ Modification
Directory maintenance	VM/Directory Maintenance (DIRMAINT)	1.2.0
Hardware error recording	Environmental Record Editing and Printing (EREP)	3.3.0
Screen manager	Interactive System Productivity Facility (ISPF)	2.2.0
System operations menus	VM/Interactive Productivity Facility (VM/IPF)	2.2.0
System control	Virtual Machine/System Product (VM/SP)	1.5.0
End-user menus	VM/Integrated System – Productivity Facility	1.5.1

Table 3. The IBM Products in VM/Integrated System Core BASE.

VM/IS Function Name	Equivalent Product Name	Equivalent Version/ Release/ Modification
Text formatter	Document Composition Facility (DCF)	1.3.1
Graphics support	Graphical Data Display Manager/Virtual Machine (GDDM/VM)	2.1.1
Presentation Graphics Support	Graphical Data Display Manager Presentation Graphic Feature (GDDM-PGF)	2.1.0
General language support routines	PL/I Transient Library	1.5.1
Background execution	VM Batch Facility	1.1.0
Shared user files	VM File Storage Facility (VM FSF)	1.1.3
Performance monitor	VM Real Time Monitor (VM RTM)	1.1.8
Performance reporting	VM/CMS Performance Monitor Analysis (VMMAP)	1.1.4

Table 4. The IBM Products in VM/Integrated System Additional BASE.

Optional Packages

Several optional packages are available with the base. These packages are listed in the following sections.

The Text Office Package (TXTO) Package

The products in the TXTO package are shown in Table 5.

Product Name	Version/ Release/ Modification
Application System (AS)	1.5.1
DisplayWrite/370	1.2.0
PL/I Resident Library	1.5.1
Professional Office System (PROFS)	2.2.2
PROFS Application Support Feature	2.2.1
PROFS Note Maintenance Facility	1.1.0
3812 Printer Support (VM3812)	1.1.2

Table 5. The IBM Products in the TXTO Package.

The Engineering/Scientific Program Development Support (E/SPDS) Package

The products in the E/SPDS package are shown in Table 6.

Product Name	Version/ Release/ Modification
Interactive System Productivity Facility/Program Development Facility (ISPF/PDF)	2.2.0
VS FORTRAN Compiler and Library (VS FORTRAN)	2.2.0

Table 6. The IBM Products in the E/SPDS Package.

The Application Development Support (ADS) Package

The products in the ADS package is shown in Table 7.

Product Name	Version/ Release/ Modification
Cross System Product/Application Development (CSP/AD)	3.1.1
Cross System Product/Application Execution (CSP/AE)	3.1.1
Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM)	1.3.0

Table 7. The IBM Product in the ADS Package.

The Transaction Support (TXN) Package

The products in the TXN package are shown in Table 8.

Product Name	Version/ Release/ Modification
Customer Information Control System/Virtual Machine (CICS/VM)	1.1.0
Common Business Oriented Language (COBOL II)	1.2.0
Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM)	1.3.0

Table 8. The IBM Products in the TXN Package.

The Data Base Query (DBQ) Package

The products in the DBQ package are shown in Table 9.

Product Name	Version/ Release/ Modification
Database Edit Facility (DBEDIT)	1.1.3
PL/I Resident Library	1.5.1
Structured Query Language/Data System (SQL/DS)	2.1.0
Query Management Facility/Virtual Machine (QMF/VM)	2.2.0
VM/System Product Interpreter Interface to SQL/DS (RXSQL)	1.1.0
Data Extract (DXT)	2.2.0

Table 9. The IBM Products in the DBQ Package.

The Intelligent Workstation Support Package

The products in the Intelligent Workstation Support package are shown in Table 10.

Product Name	Version/ Release/ Modification
IBM System/370 to IBM Personal Computer Enhanced Connectivity Facilities (ECF)	1.1.0
3270 Personal Computer File Transfer Program (3270 PC File Transfer)	1.1.0

Table 10. The IBM Products in the Intelligent Workstation Support Package.

The Networking Support (NTWK) Package

The products in the NTWK package are shown in Table 11.

Product Name	Version/ Release/ Modification
NetView	1.2.0
Remote Spooling Communication Subsystem Networking (RSCS)	2.2.0
Virtual Storage Extended/Virtual Sequential Access Method (VSE/VSAM)	1.3.0
Advanced Communications Function/Virtual Telecommunication Access Method (ACF/VTAM)	3.1.2

Table 11. The IBM Products in the NTWK Package.

The Remote Communication Support (RCS) Package

The products in the RCS package are shown in Table 12.

Product Name	Version/ Release/ Modification
Cooperative Viewing Facility (CVIEW)	2.1.1
VM/Pass-Through Facility (PVM)	1.3.0
Remote Spooling Communication Subsystem Networking (RSCS)	2.2.0
Transmisson Control Protocol/Internet Protocol for VM (TCP/IP)	1.1.1

Table 12. The IBM Products in the RCS Package.

The Support Services (SS) Package

The products in the SS package are shown in Table 13.

Product Name	Version/ Release/ Modification
VM/Distributed Systems Node Executive (DSNX)	1.1.0
VMBACKUP	1.4.1

Table 13. The IBM Products in the SS Package.

Chapter 10. What the VM/Integrated System Products Do

Chapter 9, "The VM/Integrated System Products and What They Do" on page 99 lists the products as they are packaged in VM/IS. This chapter lists the products alphabetically and briefly describes what these products can do.

ACF/VTAM: contains the interface between application programs in the host and other resources in an SNA network. It handles routing of data and allows remote terminals to connect to the host.

Application System: is a product designed to help business professionals get their jobs done effectively. Application System's extensive and easy-to-use business-oriented capabilities make it possible for managers, administrators, and staff with little or no data processing experience to:

- Collect and manage data
- Retrieve and analyze information
- Produce comprehensive business reports
- Present information graphically
- Use statistical analyses and forecasting packages
- Develop business plans and evaluate alternatives
- Manage projects with tools like network drawings and resource allocation
- Create and format letters and large documents.

CICS/VM: is a transaction processing product designed to perform a wide range of tasks, such as decision support and office systems as well as commercial transaction processing, from a single workstation. CICS/VM does the following:

- Provides a mixture of transaction-oriented and other interactive applications, such as PROFS and Application System, within a single operating system environment
- Allows the distributed systems to be managed from a central host location
- Allows sharing VSAM or SQL/DS data between CICS/VM applications on the distributed VM system.

CSP/AD: contains an interactive interface for defining, testing, maintaining, documenting and generating application programs.

CSP/AE: has the ability to execute any application defined and generated using CSP/AD.

CVIEW: is a VM-based facility that allows two to twelve users, at separate display stations, to share a single VM interactive session. CVIEW can be used for finding and correcting program errors, teaching a user to use a program or facility, editing a document interactively with the writer, or demonstrating a new program to a potential user.

DBEDIT: is a product that allows you to process data files from a data base. You can add, delete, and update records in a relational table. You can also display data from one or more data base tables.

You do not need data processing experience to use DBEDIT. Through a series of panels, you can work with a data base created by SQL/DS without having to understand the SQL/DS data manipulation commands.

DCF: is a document formatting tool that you can use to create different kinds of documents. You create DCF documents in two steps:

1. Create a source file that contains the text of the document accompanied by markup control characters that describes how you want the text arranged.
2. Format the document for printing or viewing on a specific device.

Such a formatting tool is very flexible because you can format the source files of your document for many types of printing devices.

DIRMAINT: makes it possible to make directory changes interactively by commands instead of through the use of an editor. DIRMAINT allows end-users to make changes to their own directory, such as changing their logon password. DIRMAINT also allows the system administrator to make more complex changes to the system directory, such as adding new users, enlarging existing users' minidisk storage, and deleting users from the system.

DisplayWrite/370: is an easy to use editor and formatter. It is part of the DisplayWrite family, a group of editors and formatters, used with the IBM PC, the IBM Displaywriter, the IBM System/36. and the IBM System/38. DisplayWrite/370 allows you to see immediately what your document looks like, rather than having to create a source file with markup and then formatting the document for printing.

With DisplayWrite/370, you can create, edit, print, check spelling, search for synonyms, and determine the reading grade level.

DSNX: is a program product that provides support for the central site management of a network of distributed VM/SP Systems. DSNX accomplishes this function through the distribution of VM software objects to remote locations, with minimal or no intervention at these locations.

DXT: is a program product that allows you to extract selected operational data on a periodic or one-time basis. You would be able to put extracted data into a relational data base (SQL) for easy access and reporting by products such as QMF. The extracted data may be moved from different subsystems on the same processor, between subsystems on different processors, or may be stored elsewhere as defined by the installation's needs.

ECF: provides a method of sharing resources between an IBM Personal Computer (PC) and VM/IS. With ECF, you can:

- Copy files between the PC and VM/IS
- Use VM disk space as if it were PC disk space
- Use VM files as if they were PC files
- Run VM commands or procedures from your PC
- Access VM printers from your PC
- Extract data from system databases to use on your PC
- Process information from a host data base on your PC
- Develop your own application program that uses VM/IS resources from your PC.

EREP: is a diagnostic aid intended to help IBM service representatives manage and maintain customers' data processing installations.

GDDM: is a licensed program that enables your programs to communicate with many of IBM's advanced display stations, printers, and plotters. You can write your programs to call on GDDM routines to create many types of graphics. It handles color and monochrome graphics and alphanumeric data.

GDDM-PGF: is a feature of GDDM that allows you to create business charts, such as line graphs or pie charts, for printing or for display at a display station. You do not need data processing experience to use GDDM-PGF because it allows you to draw charts interactively on a display screen.

ISPF: is a dialog manager that controls many of the interfaces of the products packaged under VM/IS. You can use the services of ISPF to develop dialogs for your applications. The dialog acts as the user interface to your application and controls the logical flow of your application. An ISPF dialog running with your application can:

- Identify choices of processing routines available to the user
- Invoke a requested routine, based on the user's choice
- Prompt the user for entry of data
- Wait for the user to enter the data
- Check the validity of the data
- Identify errors to the user
- Supply online information for the user.

You can use ISPF with your applications written in REXX, APL2, VS FORTRAN, Pascal/VS, VS COBOL II, or PL/I.

ISPF/PDF: is a menu-driven program development tool designed to help increase your productivity and reduce the time required for you to develop your programs. In particular, you can:

- Create and edit programs
- Store and retrieve your programs using libraries that you create
- Compile and execute your programs through interactive or batch facilities
- Perform general maintenance to your program libraries.

NetView: is a licensed program used to monitor an SNA network, manage it, and diagnose its problems.

PL/I Resident Library: contains a number of general support routines used by many of the products available with VM/IS. The PL/I Resident Library is required to compile PL/I programs.

PL/I Transient Library: contains a number of general support routines used by many of the products available with VM/IS. The PL/I Transient Library is required to execute previously compiled PL/I programs.

PROFS: is an administrative tool designed to aid in office and professional management. PROFS reduces the dependence on mail, telephone, and other conventional services. With PROFS, you can:

- Process schedules
- Add automatic reminders to the system
- Handle mail
- Process note and messages
- Prepare documents
- Proofread documents
- File and retrieve documents.

PROFS Application Support Feature: is a set of dialogs and programs that integrates PROFS, DW/370, and AS.

PROFS Note Maintenance Facility: is a program that helps you to manage your PROFS note logs.

PVM: allows VM/SP users to interactively access, initialize, or work on remote systems. Such a capability is useful when information on a remote system is needed on another system. Users can log on to this remote system using PVM and access the required data.

QMF/VM: is an interactive query product that allows you to create reports and charts from relational data. With QMF, you can:

- Access data kept in tables
- Perform calculations based on that data
- Produce reports in different formats
- Create and format charts
- Insert new data and change or delete existing data
- Communicate with other products.

RSCS: works in conjunction with the control program of VM/SP to control telecommunication I/O devices and lines used to automatically transfer files between:

- VM/SP users and other remote stations
- Remote stations and other remote stations
- VM/SP users and remote batch systems
- Remote stations and remote batch systems
- Remote stations and a CMS batch virtual machine.

RXSQL: is a programming interface from the VM/SP system product interpreter to the Structured Query Language/Data System. Execs written for the interpreter can use commands provided with this product to utilize the full power of the relational database model and the SQL language. SQL query, data definition, data manipulation and authorization statements can be executed. Some useful execs also supplied to do interactive SQL table queries or SQL operator command queries, execute the single SQL statement and have interactive access to the SQL help facilities.

SQL/DS: is a relational data base management system designed for both end-users and data processing professionals. It simplifies the task of handling data by providing facilities for querying data, manipulating data, and writing reports.

The commands used to manipulate the data base can be entered directly from display stations, used as utility program input, or imbedded in application programs written in languages such as VS COBOL II, PL/I, VS FORTRAN, or assembler language. In addition, RXSQL provides an interface to SQL/DS using REXX.

TCP/IP: supplies you, the user, with the capability of participating in a multi-vendor Internet network using the TCP/IP protocol set. The use of these protocols allows you to interface with other systems that have implemented the TCP/IP protocols. This connectivity includes the ability to transfer files, send mail, and log on to a remote host in a network of different systems. The network protocols supported are IBM Token-Ring, Ethernet LAN, and ProNET.

VM3812: supports the IBM 3812 page printer in a VM/SP environment. It provides a number of facilities to simplify the use of a 3812 page printer in the VM/SP environment.

VM RTM: is designed as a real-time monitor and diagnostic tool for short-term monitoring, analysis, and problem solving. You use it mainly for monitoring hardware or software to help in validating the system components and establishing requirements for additional hardware or software.

VM Batch Facility: supplies a facility for scheduling, initiating, and monitoring batch jobs that are executed within the VM/SP environment. The VM Batch Facility optionally provides for job accounting data, using standard VM/SP facilities. With the VM Batch Facility, you can:

- Monitor the job status
- Specify start run times for jobs
- Assign priorities to jobs
- Monitor active jobs to detect any that appear to be stalled
- Create EXECs to simplify the use of the facility.

VMBACKUP: is a file recovery and archive program which automates part of the management of the DASD backup process. It is used for the cyclical copying of minidisks to disk, tape or both and for performing archival storage of CMS files with emphasis placed on retention by expiration date. With VMBACKUP, you can:

- Obtain full and incremental dumps of CMS and non-CMS minidisks
- Restore a file from several levels:
  - File-level
  - Minidisk level
  - Full pack.
- Copy CMS files disk-to-disk with special file recovery
- Use VMBACKUP while users are logged on.

VM FSF: contains disk space and data file management services for users of VM/SP. These services are available within a single host environment or within a multiple-system computing network. Data and disk space may be shared, on a voluntary basis, with varying levels of protection selected by individual users. A hierarchical file directory permits users to cluster data files in a logical manner and to increase resolution in the identification of data files.

VM/IPF: is a VM/SP administrative tool designed to increase the productivity of its users. You, the user, can use IPF to perform many of the system administrative tasks involved in maintaining a computer system. The users can use VM/IPF to report problems to the system administrator, send and receive notes and messages, and generally maintain their own system accounts.

Administrators use VM/IPF to help maintain the entire VM/SP system. Through the user interface of VM/IPF, administrators can quickly and easily maintain the accounts of other users, add new users to the system, and delete users from the system. VM/IPF also assists in administering the I/O devices of the system.

VM/Integrated System – Productivity Facility: is a dialog that presents the entire VM/IS system to the user and runs under the control of ISPF and XEDIT. It is a series of panels that describe, in a task oriented fashion, products available with VM/IS. This user interface is designed to:

- Unify the products of VM/IS through a single entry point into the system
- Help a user select the correct product to complete a specific task
- Act as a learning aid to help the user become productive as quickly as possible.

The VM/IS – Productivity Facility is open-ended; you can tailor its appearance to include your applications. You can add your applications to the appropriate panels of the VM/IS – Productivity Facility to make your application appear as an integral part of VM/IS. A menu driven facility is available to help you tailor your menus.

VMMAP: supplies you with a performance analysis tool by reducing and analyzing data put out by the VM/Monitor. You can obtain reports and graphs portraying the performance and utilization of running VM/SP systems. Experienced users can use VMMAP reports to provide insight into many of the interrelationships with the VM/SP system and its performance.

VM/SP: is an IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a "real" machine.

VSE/VSAM: is used to manage files for various products such as NetView, CICS/VM and CSP.

VS COBOL II: is an IBM licensed program for use in developing business oriented applications in the VM/SP CMS environment. It is included in VM/IS because it is widely used, it interfaces with the CICS/VM, and it has full-screen interface to integrated interactive debugging capability.

VS FORTRAN Compiler and Library: is a compiled programming language especially useful for applications involving mathematical computations and other manipulations of numeric data. It is particularly suited to scientific and engineering applications.

VS FORTRAN Version 2 includes a debugging tool for your VS FORTRAN programs. It consists of a set of commands to help you locate and diagnose problems with your VS FORTRAN programs.

With this tool, you can suspend program execution, continue execution, skip sections of code, correct errors, display and set variables, set up executed input, and display output.

3270 PC File Transfer: is another communication tool for IBM PCs and VM/SP host systems. PC File Transfer allows you to:

- Create a file on your 3270 PC and send it to your personal storage on VM/IS
- Create a file on VM/IS and send it to your 3270 PC for revising or printing.

Summary of Changes

Summary of Changes
for SC24-5305-01
for VM/IS 5.1

How to Obtain the Previous Edition of This Publication:

To obtain the edition of this publication that pertains to VM/IS 5.0, order SQ24-5305.

New and Changed Information

- PFSETUP EXEC replaces the ESSSETUP EXEC.
- Discontiguous Saved Segments (DCSS) are now referred to as saved segments.
- The manual has been changed to show the updated VM/IS menu.
- The Glossary has been updated to reflect any new terms introduced.
- The manual has been updated to reflect any new system products introduced.

Miscellaneous

- The PRIMARY Menu is now called the MAIN Menu.
- This edition also includes minor editorial and technical changes.

Glossary of Terms and Abbreviations

This glossary defines terms frequently used in the *Developing An Application* series of books. For more information about the VM/IS products, see Chapter 10, "What the VM/Integrated System Products Do" on page 107. If you do not find the term you are looking for, refer to the Index or to the *IBM Dictionary of Computing, GC20-1699.*

A

A Programming Language (APL). A general purpose computing language for diverse applications.

ACF/VTAM. Advanced Communication Functions/Virtual Telecommunications Access Method.

Advanced Communication Functions/Virtual Telecommunications Access Method (ACF/VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple domain, and interconnected network capability.

Advanced program-to-program communications for VM (APPC/VM). An application program interface between two virtual machines that is mappable to the SNA LU 6.2 interface and is based on the way IUCV functions. Along with TSAF, it provides communication within a single system and throughout a collection of systems; part of VM/SP.

APL2. An IBM APL-based computing language that helps you to solve your everyday business problems interactively.

APPC/VM. Advanced program-to-program communications for VM.

Application System (AS). An IBM licensed program largely used for project management.

AS. Application System.

assembler language. A source language in which each instruction performs a single task. Assembler is one step removed from machine language.

B

BASE package. A set of system functions that provides the basic operation of VM/IS. This package is composed of several functions equivalent to products available for VM systems. This package is a prerequisite to any optional packages.

batch processing. The processing of data accumulated over a period of time in such a manner that the data so accumulated is processed in a single run. In a VM environment, batch processing is usually accomplished by one or more virtual machines installed solely for that purpose.

binary synchronous communication (BSC). A uniform procedure, using a standardized set of control characters and control character sequences for synchronous transmission of binary-coded data between systems.

BSC. Binary synchronous communication.

C

CICS/VM. Customer Information Control System for Virtual Machines.

CMS. Conversational Monitor System.

COBOL. A high-level programming language primarily for business applications. The version of COBOL that runs on VM/IS is VS COBOL II.

configuration. The definition and arrangement of devices that make up a computer system. The term can refer to either hardware or software.

Conversational Monitor System (CMS). A virtual machine operating system that provides general interactive time-sharing, problem solving, and program development capabilities, and operates only under the control of the VM/370 control program.

Cooperative Viewing Facility (CVIEW). An IBM program that allows from two to twelve users at separate display stations to view a single interactive terminal session. In addition, CVIEW supports one user acting as a consultant to another user at a different display station.

corequisite. A product that is not required to run your application but whose presence would provide additional function to your application. See also *prerequisite.*

CP. VM/370 Control Program.

Cross System Product/Application Development (CSP/AD). An interactive interface for defining, testing, maintaining, and generating applications.

Cross System Product/Application Execution (CSP/AE). A program that provides execution of applications defined by CSP/AD.

CSP/AD. Cross System Product/Application Development.

CSP/AE. Cross System Product/Application Execution.

Customer Information Control System for Virtual Machines (CICS/VM). An IBM licensed program that enables transaction processing applications to be integrated with office and commercial applications.

CVIEW. Cooperative Viewing Facility.

D

DASD. Direct access storage device.

Data Base Services Utility (DBSU). A tool provided with the SQL/DS licensed program.

Database Edit Facility (DBEDIT). A facility for quick and easy access to relational data base tables through a series of prompting screen panels. These panels provide full-screen data retrieval, updating, inserting, and deleting capabilities.

Data Extract (DXT). An IBM licensed program that allows you to extract selected operational data.

Data Space/Sharing Facility (DSSF). A facility used in VM FSF.

DBEDIT. Database Edit Facility.

DBSU. Data Base Services Utility.

DCF. Document Composition Facility.

dialog. A series of panels that prompts the customer for information. The panels use the information to lead the customer to a particular application required to perform a specific task.

Direct access storage device. Disk devices used to store files and data.

DIRMAINT. VM/Directory Maintenance.

DisplayWrite/370. A host-based text editor and formatter used to create and revise many types of documents. An optional image and graphics feature is available with the program.

DMKBOX. System logo file. A file used to define the system logo.

DMKRIO. Real input/output (I/O) configuration file. A file used to define the peripheral I/O devices of a VM/IS operating environment.

DMKSNT. System name table file. A table used to allow the sharing of program segments between virtual machines. The table contains the name and location of saved systems, including discontiguous shared and non-shared segments.

DMKSYS. CP system control file. A file used to define the CP system residence disk, the real storage size, the CP-owned DASD volumes, the VM/370 system operator's user ID, the system timer value, and other system variables.

Document Composition Facility (DCF). A text formatting package, designed to be used on documents of any size.

DSNX. Virtual Machine/Distributed Systems Node Executive.

DSSF. Data Space/Sharing Facility.

DXT. Data Extract.

E

ECF Servers-Requesters. Enhanced Connectivity Facility Servers-Requesters.

Enhanced Connectivity Facility Servers-Requesters (ECF Servers-Requesters). An IBM licensed program that allows personal computer users to access and work with host computing systems.

Environmental Record Editing and Printing (EREP). A hardware error tracking and recording program operating under VM/SP.

EREP. Environmental Record Editing and Printing.

EXEC procedure. A CMS function that allows you create new commands by setting up frequently used sequences of CP commands, CMS commands, or both, together with conditional branching facilities, into files with a file type of EXEC to eliminate the repetitious rekeying of those command sequences. When you enter the file name of the EXEC procedure, the commands in the file are executed.

F

feature. A part of an IBM product that may be ordered separately.

Feature program product (FPP). A tape format used to package your application programs.

flat file. A flat file is one that uses only two dimensions. The first dimension is the record length (the number of spaces plus characters in a line), giving you the width of the file. The second dimension is the number of records in the file, giving you the length of the file. An example of a flat file is a text file. You can

normally display the contents of a flat file on a terminal screen. When stored on a minidisk, a flat file can be stored in either fixed format (that is, the entire file is as wide as the longest record; shorter records are filled with blanks), or in variable format (that is, each record takes up only the amount of room used by its contents; this format takes less space).

FORTRAN. A programming language primarily designed for applications involving numeric computations. The version of FORTRAN that runs under VM/IS is VS FORTRAN Version 2.

FORTRAN Utilities. A set of utility programs (included in the VS FORTRAN Compiler and Library) that provide useful functions for your FORTRAN programs. These utilities allow you to perform such tasks as executing VM/SP commands from your program, reading the system time and date, and clearing an IBM 3270 terminal display.

FPP. Feature program product.

fullscreen. Applications that display an entire screen of information at a time. These screens include selection menus, data entry panels, and help panels.

G

GCS. Group control system.

GDDM. Graphical Data Display Manager.

GDDM-PGF. Graphical Data Display Manager with the Presentation Graphics Feature.

Generalized Markup Language (GML). A set of control words (tags) interspersed with actual document text and used with DCF to format documents.

GML. Generalized Markup Language. Used with DCF to format documents.

Go facility. A feature included in VM/IS by which a customer can bypass the normal menu flow to access a specific panel or product. A customer can use a special menu (the GO MENU), can enter a keyword, or can enter an option number to access the desired panel or product.

Graphical Data Display Manager (GDDM). A graphics package that you can use to produce graphics for many of IBM's advanced display stations, printers, and plotters.

Graphical Data Display Manager with the Presentation Graphic Feature (GDDM-PGF). A graphics package that you can use to produce many different types of charts and graphics.

Group control system (GCS). A component of VM/SP that provides simulated MVS services and unique supervisory services to help support a native SNA network.

I

IBM BASIC/VM. An IBM BASIC-based programming language. BASIC stands for *Beginner's All-purpose Symbolic Instruction Code*.

Indexed Sequential Access Method (ISAM). An access method used to directly retrieve or update particular blocks of a data set on a direct access device, using an index to locate the data set.

INSTFPP EXEC. An EXEC program supplied with VM/IS that is designed to install applications, programs, or packages from one or more tapes when the tapes use FPP (feature program product) format.

Interactive Problem Control System (IPCS). A component of VM/370 that permits online problem management, interactive problem diagnosis, online debugging for disk-resident CP abend dumps, problem tracking, and problem reporting.

Interactive Structured Query Language (ISQL). A facility of SQL/DS that provides online data query and report writing support.

Interactive System Productivity Facility (ISPF). A facility used for defining and controlling interface dialogs. These dialogs are used extensively in developing user interfaces and online tutorial sessions for applications.

Interactive System Productivity Facility/Program Development Facility (ISPF/PDF). A menu-driven program development tool to help you efficiently create and maintain programs. It allows you to create, edit, compile, execute, and store your programs.

Inter-User Communication Vehicle (IUCV). A CP system service used to send information from one virtual machine to another.

IPCS. Interactive Problem Control System.

ISAM. Indexed Sequential Access Method.

ISPF. Interactive System Productivity Facility.

ISPF/PDF. Interactive System Productivity Facility/Program Development Facility.

ISQL. Interactive Structured Query Language.

IUCV. Inter-User Communication Vehicle.

L

line mode. Pertaining to applications that display or accept information one line at a time.

load-and-go. A method of packaging an application or a system in which the application or system has been pretailored to suit most customer's needs. Generally, a customer can use the application or system immediately after loading it.

M

menus. Panels consisting of a list of items from which the customer can select one item. The chosen item could lead to another more detailed menu, a panel requesting information, or directly into an application that the customer can use to complete a task.

minidisk. All, or a logical subdivision of, a physical disk storage device that has its own address, consecutive storage space for data, and an index or description stored so that data can be analyzed; synonym for virtual disk.

migrating. The process of installing a higher level of the system or of one or more individual products without losing or reentering associated data. The process replaces changed system or application files and, in some cases, changes the format of existing data to be compatible with the new system or application.

mirror-disk. Another disk or minidisk of the same logical size as a disk or minidisk that contains critical data. The application places copies of all files contained on the main disk or minidisk onto the mirror-disk on a regular basis, either each day or each week on an automatic basis, or whenever necessary under the control of the administrator.

N

NAMESYS. Macro included in the system name table (DMKSNT) assembler file.

NetView. A licensed program used to monitor an SNA network, manage it, and diagnose its problems.

O

optional package. One or more application programs grouped according to related function and designed to be installed on the VM/IS BASE package. Each package is pretailored to be installed quickly and easily and to become an integrated part of VM/IS with a minimum of work.

P

package. One or more application programs or functions grouped according to function and provided to the customer as a pretailored system. VM/IS is composed of a BASE package and several optional packages.

packaging. The procedure and format used to put a copy of your application or a group of related applications onto a tape. The package includes documentation to install, customize, maintain, and use the application or group of applications.

panel. A complete set of information shown in a single display on a display station screen. See also *screen*.

panel tailoring files. Files created by the user of an application to modify or customize the VM/Integrated System – Productivity Facility. These files are created to add one or more items to an existing menu or to add another complete menu or panel to the existing series of menus or panels.

Pascal/VS. An IBM Pascal-based programming language compiler.

PC. Personal computer.

personal computer (PC). A desk-top, floor-standing, or portable microcomputer designed primarily to give independent computing power to a single user.

PF keys. Program function keys. A set of keys on your terminal that can be programmed to perform specific commands when pressed.

PL/I (programming language/one). A programming language designed for numeric scientific computations, business data processing, systems programming, and other applications.

PL/I Resident Library. A collection of general support routines used by many of the products available through VM/IS and required to compile PL/I programs.

PL/I Transient Library. A collection of general support routines used by many of the products available through VM/IS and required to execute previously compiled PL/I programs.

prerequisite. A product that is required before you are able to run your application. See also *corequisite*.

processor. The part of the computing device that performs the actual data manipulation.

Professional Office System (PROFS). An office product that you can use to manage many office tasks such as sending messages to other VM/IS users, maintaining your personal calendar, and preparing documents.

PROFS. Professional Office System.

PROFS Notelog Maintenance. An IBM licensed program to keep note logs up to date.

pseudo-code. A form of coding that closely resembles executable code, but which must be translated into formal code prior to execution. Pseudo-code can simplify both the testing of logic and the actual coding.

PVM. VM/Pass-Through Facility.

Q

QBE. Query-by-example.

QMF/VM. Query Management Facility for VM.

Query Management Facility for VM. An IBM licensed program designed to provide interactive facilities to users with little or no data processing background.

Query-by-example (QBE). A data query language provided in the IBM licensed program QMF/VM.

R

Remote Spooling Communication Subsystem (RSCS). A special-purpose subsystem that supports the transmission, relaying, and reception of messages, commands, files, and jobs over a computer network.

Restructured Extended Executor Language (REXX). An EXEC language used in System Product Interpreter programs.

REXX. Restructured Extended Executor Language.

RSCS. Remote Spooling Communication Subsystem.

RXSQL. The System Product Interpreter interface between REXX and SQL/DS.

S

SAM. Sequential access method.

saved segment. A characteristic of a saved system whereby one or more segments of reentrant code stored in real storage can be shared by several virtual machines. This can reduce real storage usage when multiple users work with an application concurrently. A saved segment is sometimes called a *shared segment*.

screen. A physical display surface. See also *panel*.

SCRIPT/VS. A text processing program; part of DCF.

sequential access method (SAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or direct access device.

Servers-requesters programming interface (SRPI). Part of the ECF Servers-Requesters package that allows you to develop your own servers and requesters.

SNA. Systems Network Architecture.

SQL/DS. Structured Query Language/Data System.

SRPI. Servers-requesters programming interface.

Structured Query Language/Data System (SQL/DS). An IBM licensed program that manages relational data bases.

Systems Network Architecture (SNA). A network communications protocol used between peripheral I/O devices, processors, and applications.

System Product Editor. Synonymous with XEDIT.

SYSVAR. System variables that can be used to control the processing of a document in SCRIPT/VS.

T

task-oriented. Pertaining to a method of guiding the customer to the product most suited to perform a specific job starting from a list of general tasks. Conventional (non-task-oriented) methods involve starting from a list of products, forcing the customer to define the task to fit the product.

TCP/IP. Transmission Control Protocol/Internet Protocol.

Technical newsletter (TNL). An update to an existing manual (usually replacement pages and an updated table of contents).

TNL. Technical Newsletter.

transactional processing. A system of processing whereby a device (usually a user's virtual machine) submits a job to another device (this could be a service machine) for processing, after which the processing device returns information, data, or acknowledgment to the originating device.

Transmission Control Protocol/Internet Protocol (TCP/IP). An interface to transfer files, mail and user's jobs to a remote host in an Internet network.

Transparent Services Access Facility (TSAF). A VM/SP communications tool that lets users connect to and communicate with local or remote virtual machines within a collection of as many as eight systems.

TSAF. The Transparent Services Access Facility.

U

USERSCOM. A segment of the VM Batch Facility that can be tailored to fit the local environment.

USERSJOB. A segment of the VM Batch Facility that can be tailored to fit the local environment.

USERSRJE. A segment of the VM Batch Facility that can be tailored to fit the local environment.

V

VDISK. Virtual disk.

virtual disk. See *minidisk*.

virtual machine. A functional equivalent of an IBM System/370 computing system. Many virtual machines can run on a single real computer under the VM/SP operating system. Each virtual machine appears to the user to be a fully functional computer, with its own card reader, card punch, printer, and disk space.

Virtual Machine Monitor Analysis Program (VMMAP). A performance analysis tool that generates reports and graphs that help you portray the performance and utilization of a running VM/SP system.

Virtual Machine/Distributed Systems Node Executive (VM/DSNX). An IBM licensed program for the distribution and implementation of changes on remote systems in a distributed systems environment.

Virtual Machine/System Product (VM/SP). The operating environment of VM/IS; an IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist.

Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM). An access method for direct or sequential processing of fixed and variable length records on direct access devices.

Virtual Telecommunications Access Method (VTAM). A set of programs that control communication between display stations and application programs running under VM/SP. See also *ACF/VTAM*.

VM Batch Facility. A licensed program that allows you to submit and control batch jobs under VM.

VM File Storage Facility (VM FSF). A licensed program that controls the sharing of files among virtual machines.

VM FSF. VM File Storage Facility.

VM/Real Time Monitor (VM/RTM). A real-time monitor and diagnostic tool for short term monitoring, analysis, and problem solving. It is used mainly for monitoring hardware or software to help in validating the system components and establishing requirements for additional hardware or software.

VMBACKUP Management System. A licensed product that performs CMS and non-CMS backup and restore functions. The VM Archive Subsystem of VMBACKUP-MS provides the capability to archive files and retrieve them when needed.

VMMAP. Virtual Machine Monitor Analysis Program.

VM3812. 3812 Pageprinter Support.

VM/Directory Maintenance (DIRMAINT). A utility program that is used to maintain the system directory.

VM/DSNX. Virtual Machine/Distributed Systems Node Executive.

VM/Integrated System (VM/IS). A computing system for business and engineering/scientific professionals, designed for use with the IBM 43xx and IBM 937x families of processors.

VM/Interactive Productivity Facility (VM/IPF). An IBM licensed program that is used both by end-users and by system administrators to perform routine operations such as sending messages, notes, and documents, maintaining the system directory, and reporting and managing problems.

VM/IPF. VM/Interactive Productivity Facility.

VM/IS. VM/Integrated System.

VM/IS – Productivity Facility. The interface to products and functions available through VM/IS.

VM/Pass-Through Facility (PVM). A facility that allows you to log on and use your VM/IS system from a remote system in your computer network or to log on and use a remote system from your VM/IS system.

VM/RTM. VM/Real Time Monitor.

VM/SP. Virtual Machine/System Product.

VM/370 Control Program (CP). A component of VM/SP that controls the resources of the real machine so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent to an IBM System/370.

VS COBOL II. An IBM COBOL-based programming language compiler that can run under VM/IS.

VS FORTRAN. An IBM FORTRAN-based programming language compiler with a subroutine library.

VS FORTRAN IAD. VS FORTRAN Interactive Debug.

VS FORTRAN Interactive Debug. An interactive debugging tool for your VS FORTRAN programs included as part of VS FORTRAN.

VSAM. VSE/Virtual Storage Access Method.

VSE/VSAM. Virtual Storage Extended/Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method.

X

XEDIT. The System Product Editor; part of VM/SP.

Numerics

3270 PC File Transfer. 3270 Personal Computer File Transfer Program. An application that allows you to create a file on your 3270 PC and send it to your personal storage on VM/IS, or to create a file on VM/IS and send it to your 3270 PC for revision or printing.

3812 Pageprinter Support. An interface to the IBM 3812 Pageprinter that simplifies sending a print job to the printer.

Index

A

acceptable responses 57
ACF/VTAM 107
  brief description 107
  definition 117
  documenting tool 94
acronyms defined 87
adding help panels 54
adding online introduction topic 54
adding your application to VM/IS
  PFSETUP rebuild process 10
ADDRESS COMMAND statement 71, 74
ADMIN user ID 77
administering VM/IS 8
administration documentation
  administering 85
  customizing 85
  file contents 86
  glossary 87
  installing 85
  introduction 85
  migrating 86
  operating 85
  servicing 86
  suggestions 84
APL2 25
  definition 117
  discussion 51
  using editors 53
  with ISPF 109
application files 67
application identifier file 66, 70
  contents 70
  sample 70
application installation EXEC 66
application packaging requirements
  VM/IS considerations 34
  VM/SP considerations
    component code 34
    DASD layout 43
    migration 43
    minidisks 43
    saved segments 44
    service machines 46
    SHUTSERV EXEC 46
    SNTMAP EXEC 45
    SYSTEM PROFILE 46
application structure 42
Application System 107
  brief description 107
  definition 117
  documenting tool 92, 93, 94
  in scenario 15
application, invoking 35
application, testing
  debugging aids and tools 59
  documenting 58, 59
  for performance 59
  functional 60
  functional interaction 59
  guidelines 57
  keeping records 57
  methods 57
  order of testing 58
  plan 58
  purpose of testing 57
  test cases and data 58
  testing the documentation 60
  testing the system 60
  tools and debugging aids 58
  usability 60
  using experienced testers 60
  when to test 57
  who should test 57
AS 30
  See also Application System
assembler language 25, 49
  definition 117
  discussion 50
  with SQL/DS 111

B

backup, system 29
batch processing
  considerations 21
  planning 23
building your tape 75

C

calling subroutines 52
cases, test 58, 59
central message facility 40
changes, project 53
  affect on code 53
  affect on documentation 53
  handling 53
choosing additional minidisks 36
CICS/VM
  brief description 107
  definition 118
  documenting tool 93
CMS commands 52
CMS EXEC 77
  system restrictions 77

CMS macros 51
CMSIUCV macro 26
CMSUSER user ID 74
codes, return
  documenting 91
  0 72
  111 73
  777 72
  888 73
  999 73
coding summary 55
command documenting 40
command syntax definition language 40
command-driven interface 21
commands 40
commands and EXECs
  CMS 77
  CMSIUCV 26
  CONSOLE 26
  DIAGNOSE 26
  DMSFREE 26
  DMSKEY 26
  GETFMADR 36
  HNDIUCV 26
  installation 63, 64, 66, 67
  INSTFPP 64, 66, 69, 70, 88
  INSTFPP EXEC 69
  LINERD 26
  LINEWRT 26
  LISTFILE 77
  NAMESYS 86
  PFSETUP 10, 55, 91
  profile 68, 86, 87, 91
  RDTERM 26
  SHUTSERV 46
  SNTINFO 72
  SNTMAP EXEC 45
  SYSPROF EXEC 27
  SYSTEM PROFILE 46
  tape build EXEC 75
  verification 64, 68, 74, 75, 91
  verification EXEC 74
  VMFPLC2 75
  WRTERM 26
communicating with other applications 27
communicating with other systems 27
communicating with the operating system 27
communicating with the user 27
component code 34
  hints for choosing 34
configuration 4
CONSOLE macro 26
control tools 53
corequisites 85
corequisites of planning 24
CP facilities
  IUCV 26
  logical device support facility 26
CP facilities *(continued)*
  message system service 26
  single console image facility 26
  TSAF 27
  using 26
CSP/AD
  brief description 107
  definition 117
CSP/AE
  brief description 107
  definition 117
customizing files 43
customizing VM/IS products 42
CVIEW
  brief description 107
  definition 117
  documenting tool 94

D

DASD
  adding 43
  planned 23
  system volumes 43
  tailoring 43
DASD layout, tailoring 43
DASD SNTMAP 45
data and file structure
  designing 34
  planning 24
data management tools
  CICS/VM 107
  DBEDIT 107
  ECF 108
  QMF/VM 110
  RXSQL 110
  SQL/DS 111
  VM/SP 112
data recovery 12, 28
  interrupted session 28
  mirror-disk 29
  VMBACKUP 29
data, test 58, 59
DBEDIT
  brief description 107
  definition 118
  documenting tool 93
DCF
  adding tags 55
  brief description 108
  definition 118
  documenting tool 93
  tailoring PROFS 55
debugging aids 58, 59
default information 39
defining errors 57
defining I/O devices 23

defining sharable code segments 23
defining system macros 23
defining terms, acronyms 87
designing summary 47
designing the application program 42
designing your publications 82
  for the administrator 82
  for the end-user 82
development stages 9
DIAGNOSE instruction 26
dialog 5, 6, 8
dialog control tools
  GDDM 54
  ISPF 12, 35, 54, 109
  REXX 50
  VM/IS – Productivity Facility 112
dialog management 35
directory entry
  changing 45
  sample 89
directory entry description 89
directory, system
  tailoring 45
DIRMAINT
  brief description 108, 122
DisplayWrite/370
  brief description 108
  definition 118
  documenting tool 92, 93
DMKFCB 44
  tailoring 44
DMKRIO 44
  definition 23, 118
  supplying information 88
  tailoring 44
DMKSNT 45
  definition 23, 118
  generating saved segment 72
  preinstallation requirements 70
  Saved Segment 44
  SNTINFO 72
  specifications 86
  supplying information 88
  tailoring 44
DMKSYS 44
  definition 23, 118
  tailoring 44
DMSFREE macro 26
DMSKEY macro 26
documentation 33, 81
  content 81
  help panels 81
  installation information 81
  manual 81
  physical appearance 81
  planning 21
  recovery strategy 23
  source listings 81
documentation *(continued)*
  types of manual 81
  writing 83
documentation as a manual 81
documentation as help panels 81
documentation guidelines 88
documentation tools
  administrative 94
  batch facilities 94
  communications 94
  editors 92
  file management 93
  formatters 93
  graphics support 93
  personal computer support 93
  printer support 93
documentation, testing 60
documenting summary 94
DSNX
  brief description 108
  definition 122
DTRFDEF function 35
DXT
  brief description 108
  definition 118

E

ease-of-use, testing 57
ECF
  brief description 108
  definition 118
editors 53
  ISPF/PDF 53
  XEDIT 53
end-user documentation
  examples 84
  glossary 84
  important tasks 84
  introduction 84
  suggestions 83
enrolling new users 85
entry, directory
  changing 45
  sample 89
EREP
  brief description 108
  definition 118
error messages 40
errors, defining 57
establishing a schedule 29
example directory entry 88
example tape build EXEC 75
EXECs and commands
  CMS 77
  CMSIUCV 26
  CONSOLE 26
  DIAGNOSE 26

EXECs and commands *(continued)*
DMSFREE 26
DMSKEY 26
GETFMADR 36
HNDIUCV 26
installation 63, 64, 66, 67
INSTFPP 64, 66, 69, 70, 88
INSTFPP EXEC 69
LINERD 26
LINEWRT 26
LISTFILE 77
NAMESYS 86
PFSETUP 10, 55, 91
profile 68, 86, 87, 91
RDTERM 26
SHUTSERV 46
SNTINFO 72
SNTMAP EXEC 45
SYSPROF EXEC 27
SYSTEM PROFILE 46
tape build EXEC 75
verification 64, 68, 74, 75, 91
verification EXEC 74
VMFPLC2 75
WRTERM 26
executing CMS commands 38
explanation 85

F

facility, parsing 40
familiarizing new users 38
fast-path 38
file and data structure
designing 34
planning 24
file contents 86
FILEDEF command 35
format, package tape 64
FORTRAN UTILITIES
definition 119
functional testing 60

G

GDDM
brief description 109
definition 119
documenting tool 93
GDDM as dialog manager 54
GDDM-PGF
brief description 109
definition 119
documenting tool 93
Generalized Markup Language 55
GETFMADR 36
GML 55
Go Facility 8, 38
definition 119
graphics tools
GDDM 109
GDDM-PGF 109

H

handling project changes 53
help function 37
help panels 8, 42
hierarchy of commands 27
hierarchy, menu 9
HNDIUCV macro 26

I

IBM BASIC/VM 25
definition 119
discussion 51
IBM Displaywriter 108
IBM System/36 108
IBM 3812 page printer 111
identifier file 66, 70
contents 70
naming convention 66
sample 70
information about minidisk links 85
information messages 40
input/output routines
using CMS commands 26
using EXEC2 26
using REXX 26
using XEDIT 26
installation documentation 4
installation EXEC 63, 66, 67
guidelines 70
naming convention 67
installation EXEC prompt 69
installation information 81
installation messages 72
installation process 69
installation, testing 78
installing VM/IS
installation documentation 4
installing VM/IS 4
preconfigured system 4
predesigned software 4
INSTFPP EXEC 64, 66, 69, 70
Inter-User Communication Vehicle 47
Interactive System Productivity Facility 35
interactive tutorial session 39
interfaces
command-driven 27, 34, 54
designing 34
menu-driven 54
panel-driven 27, 34, 37
tailorable 26

interfaces *(continued)*
using GDDM 54
using ISPF 54
VM/IS – Productivity Facility 27
internal communications in your application 47
invoking your application 35
ISPF
brief description 109
debugging aids 59
definition 119
documenting tool 92
in scenario 12
instructions for tools 87
ISPF as dialog manager 54
ISPF dialogs 35
ISPF/PDF
brief description 109
definition 119
documenting tool 92, 93
in scenario 16
program development tool 52
using editors 53
IUCV 14, 47

L

languages and tools
APL2 25
discussion 51
assembler 25, 49
discussion 50
IBM BASIC/VM 25
discussion 51
ISPF 43
ISPF/PDF 109
Pascal/VS 25
discussion 51
PL/I Resident Library 109
PL/I Transient Library 109
REXX 25, 43, 49, 52
discussion 50
RXSQL 52
SQL/DS 52
VM FSF 111
VS COBOL II 25, 112
discussion 50
VS FORTRAN 25, 49, 112
discussion 50
XEDIT 43
learning VM/IS 5
LIBDEF service 35
library option 41
line-driven application 34
LINERD macro 26
LINEWRT macro 26
link addresses 71
LISTFILE command 77
load-and-go 90
definition 120
logical tape file
definition 66
description 66
tape file n 68
tape file 1 66
tape file 2 67, 87
tape file 3 67
tape file 4 67
tape file 5 67
tape file 6 68
tape layout
illustration 65
tape mark 66

M

MAIN MENU 7
hierarchy 7
library option 41
task-oriented 7
VM/IS MAIN MENU 7
MAINT DIRECT file 44
managing your project 29
manual tailoring 91
manual verification 91
Memo to Users 67, 75, 87
corequisites 90
installation information 90
messages 91
preinstallation information 88
prerequisites 90
printing 75
servicing 92
special instructions 92
suggestions 87
tailoring 91
verification 91
MEMORY SNTMAP 45
menu 3, 4, 5, 6, 7
message interface 40
message interface numbering convention 40
messages, installation 72
methods of communication
BSC 47
IUCV 47
RSCS 47
SNA 47
TSAF 47
VMCF 47
VTAM 47
with other systems 47
migration 42, 43, 86
minidisks
DIRMAINT 195 77
enlarging 44
MAINT 001 (temporary) 69

minidisks *(continued)*
MAINT 002 (temporary) 69
MAINT 19E 46, 71, 73, 74
MAINT 191 69, 71
MAINT 193 45, 64, 69, 70, 71
MAINT 295 45
MAINT 31A 90
MAINT 319 69, 71
MAINT 326 44
passwords 46
minimum virtual storage 88
mirror-disk 12, 29
multiple features 64, 67

N

naming convention
application identifier file 66
application installation EXEC 67
memo to users 67
verification EXEC 68
naming conventions, conflicts 61
national language support 83
considerations 83
NetView
brief description 109
definition 120
networking and communications tools
ACF/VTAM 107
CSP/AD 107
CSP/AE 107
CVIEW 107
DSNX 108
DXT 108
NetView 109
PVM 110
RSCS 110
TCP/IP 111
VSE/VSAM 112
3270 PC File Transfer 113
new MAIN menus 41

O

objectives, setting 21
of messages.message explanation 85
online help 38
for information panels 39
for menus 38
for work panels 39
online introduction 5, 41
hierarchy 5
subtopic-selection menu 6
topic-selection menu 6
open-ended system 3, 4
OPTPK1 43
order of testing 58
OS PL/I V2 25
output and input routines
using CMS commands 26
using EXEC2 26
using REXX 26
using XEDIT 26

P

package tape format 64
packages
ADS 102
DBQ 103
definition of 3
E/SPDS 101
Intelligent Workstation Support 103
NTWK 104
RCS 104
SS 105
TXN 102
TXTO 101
VM/IS BASE 99
packaging your application 63
definition 63
documenting 63
panel definition files 41
panel identification 36
panel layout 36
panel tailoring files 5, 10, 41, 44, 68
panel-driven application 34
panel-driven interface 21
parsing facility 40
parsing facility, DLCS 40
Pascal/VS 25
definition 120
discussion 51
with ISPF 109
password information 92
passwords 71
passwords on minidisks 46
PF keys 35, 36, 37
PFSETUP EXEC 10, 55
in the Memo to Users 91
planning summary 30
plan, test 58
PL/I Resident Library
brief description 109, 120
subroutine library 51
used in coding 51
PL/I Transient Library
brief description 109, 120
subroutine library 51
used in coding 51
PL/I with ISPF 109
PL/I with SQL/DS 111
preconfigured system 4
predefined routines 51

predesigned software 4
prerequisites 85
prerequisites of planning 24
pretailoring your application 36
problem diagnosis 85
problem handling 85
prod level file 69
products in VM/IS
  AS 30
  DCF 55
  ISPF 35, 53, 59
  ISPF/PDF 52, 53
  PL/I Resident Library 51
  PL/I Transient Library 51
  PROFS 30, 36, 37, 55
  PVM 26
  QMF/VM 52
  RSCS 47
  SQL/DS 21, 24, 34, 52
  VM Batch Facility 46
  VMMAP 44
  VM3812 25
  VM/IPF 22, 45, 77
  VS FORTRAN 25, 58
  VTAM 47
professional management and document formatting
  Application System 107
  DCF 108
  DisplayWrite/370 108
  PROFS 110
  PROFS Application Support Feature 110
  PROFS Note Maintenance Facility 110
  VM3812 111
profile EXEC 68
PROFPK 43
PROFS
  brief description 110
  definition 120
  documenting tool 93, 94
  in scenario 15
  PF key function 36
  tailoring 55
  using 30
PROFS Application Support Feature
  brief description 110
  documenting tool 93
PROFS Note Maintenance Facility
  brief description 110
  definition 121
  documenting tool 93
program development tools 52
program directory information 88
programming languages and tools
  APL2 25
    discussion 51
  assembler 25, 49
    discussion 50
  IBM BASIC/VM 25
    discussion 51
programming languages and tools *(continued)*
  ISPF 43
  ISPF/PDF 109
  Pascal/VS 25
    discussion 51
  PL/I Resident Library 109
  PL/I Transient Library 109
  REXX 25, 43, 49, 52
    discussion 50
  RXSQL 52
  SQL/DS 52
  VM FSF 111
  VS COBOL II 25, 112
    discussion 50
  VS FORTRAN 25, 49, 112
    discussion 50
  XEDIT 43
project changes 53
  affect on code 53
  affect on documentation 53
  handling 53
project management 29
project management tools 53
pseudo-code 15, 58
publications, writing
  administration documentation 84
  end-user documentation 83
  Memo to Users 87
purpose of testing 57
PVM
  brief description 110
  definition 122
  documenting tool 94

Q

QMF/VM
  brief description 110
  definition 121
  interacting with SQL/DS 52

R

RDTERM macro 26
recovery 28
  interrupted session 28
  mirror-disk 29
  VMBACKUP 29
recovery strategy 23
  documenting 23
  planning 23
reference manual 82
results of testing, documenting 58
return codes 69
  documenting 91
  0 72
  111 73
  777 72

return codes *(continued)*
888 73
999 73
REXX 16, 25, 49, 70, 74, 87
discussion 50
interface 52
with ISPF 109
with SQL/DS 111
RSCS
brief description 110
definition 121
documenting tool 94
RXSQL
brief description 110
definition 121
interface with SQL/DS 52

S

sample directory entry 89
saved segment 21, 27, 44, 70, 72, 86, 90
saved segments 44
scenario 11
schedules 53
scheduling 29
screen flow 13
security 28
password 28
security level 28
service 92
service machines
AUTOLOG1 46
communications 47
in scenario 12
instructions to initialize 85
linking instructions 87
loading files to 71
sample information 88
SYSDUMP1 29
using 46
setting objectives 21
planning 23
shutdown information 85
SHUTSERV EXEC 46
SHUTSERV TABLE 46
skills needed for service 86
SNTINFO 72
SNTMAP EXEC 45
DASD SNTMAP 45
MEMORY SNTMAP 45
source code files 67
source listings 81
space requirements 88
specifications measurable 57
spool state parameters 71, 74
SQL/DS
brief description 111
definition 121
SQL/DS *(continued)*
in scenario 12, 14
interacting with 52
stages in development 9
starting the ISPF environment 35
startup information 85
statement, ADDRESS COMMAND 71, 74
step-by-step format 84, 86
storage space
adding 43
planned 23
system volumes 43
tailoring 43
subtopic-selection menu 6
successful installation, testing for 78
summary
coding 55
designing 47
documenting 94
planning 30
SYSPROF EXEC 27
system administration tools
DIRMAINT 108
EREP 108
VM Batch Facility 111
VM RTM 111
VMBACKUP 111
VMMAP 112
VM/IPF 112
system architecture 22
definition 22
system backup 29
system configuration 23, 43
system directory 23
definition 23
tailoring 45
system name table
system name table, updating 45
system product interpreter 70
SYSTEM PROFILE 46
system profile EXEC 27, 45
system restrictions 77
system, testing 60

T

tailorable interfaces 26
system profile EXEC 27
VM/IS – Productivity Facility 27
tailored MAIN menus 41
tailoring DASD layout 43
tailoring existing products 55
tailoring the system 43
tailoring VM/IS 4
tailoring your application 36
tape
building your tape 75
layout 64

tape *(continued)*
using the VMFPLC2 command 75
tape layout
description 66
illustration 65
tape mark 66
task-oriented 7, 82
task-oriented format 41
tasks in development 9
TCP/IP
brief description 111
definition 121
documenting tool 94
temporary minidisks 69
terms defined 87
testing for performance 59
testing for successful installation 78
testing your application
debugging aids and tools 59
documenting 58, 59
for performance 59
functional 60
functional interaction 59
guidelines 57
keeping records 57
methods 57
order of testing 58
plan 58
purpose of testing 57
test cases and data 58
testing the documentation 60
testing the system 60
tools and debugging aids 58
usability 60
using experienced testers 60
when to test 57
who should test 57
text editors 53
ISPF/PDF 53
XEDIT 53
tools, control 53
tools, documentation
administrative 94
batch facilities 94
communications 94
editors 92
file management 93
formatters 93
graphics support 93
personal computer support 93
printer support 93
tools, project management 53
topic-selection menu 6
transactional processing 12, 21
transferring data between products 24
translation considerations 83
Transparent Services Access Facility 27, 47
TSAF 27, 47
tutorial Information 38, 39
two or more programming languages 52
types of manual 81
administration primer 82
end-user primer 82
Memo to Users 82
suggested names 82

U

updating the system name table 45
usability testing 60
USER DIRECT 23
USER DIRECT file 77
user IDs
AUTOLOG1 46
communications 47
in scenario 12
instructions to initialize 85
linking instructions 87
loading files to 71
sample information 88
SYSDUMP1 29
using 46
user interfaces
command-driven 27, 34, 54
designing 34
menu-driven 54
panel-driven 27, 34, 37
tailorable 26
using GDDM 54
using ISPF 54
VM/IS – Productivity Facility 27
using error messages in testing 59
using text editors 53
ISPF/PDF 53
XEDIT 53
utility languages 51

V

verification EXEC 68, 74, 75
creating 74
verification EXEC guidelines 74
virtual link addresses 71
virtual machine
definition 122
SYSDUMP1 29
understanding 22
Virtual Machine Communication Facility 47
virtual storage 23, 88
default amount 23
VM Batch Facility
brief description 111
definition 122
documenting tool 94
service machines 46

VM FSF
brief description 111
definition 122
documenting tool 93
VM RTM
brief description 111
definition 122
performance tool 59
VMBACKUP
brief description 111
definition 122
recovery tool 29
VMCF 47
VMFPLC2 command 75
example of use 76
VMMAP
as an example 44
brief description 112
definition 122
performance tool 59
VMSPK1 43
VMSRES 43
VM3812
brief description 111, 123
documenting tool 93
using 25
VM/IPF
administrative tool 22
brief description 112
definition 122
to access USER DIRECT 77
to update DMKSNT 45
updating CP directory 90
VM/IS concepts
administering VM/IS 8
learning VM/IS 5
tailoring VM/IS 4
VM/IS considerations
choosing additional minidisks 36
DTRFDEF function 35
FILEDEF command 35
LIBDEF service 35
starting the ISPF environment 35
VM/IS MAIN MENU 7
VM/IS – Productivity Facility
adding help panels 54
adding online introduction topic 54
brief description 112
customizing 54
definition 122
help panels 42
new MAIN menus 41
online introduction 41
tailored MAIN menus 41
tailoring 41
VM/SP
brief description 112
central message facility 40
VM/SP *(continued)*
debugging aids 59
definition 122
DMKFCB 44
DMKRIO 44
DMKSNT 44
DMKSYS 44
IUCV 26, 47
logical device support facility 26
message system service 26
parsing facility 40
single console image facility 26
SNTMAP 45
TSAF 27, 47
VMCF 47
windowing 38, 40
VS COBOL II 25, 49
brief description 112
definition 122
discussion 50
in scenario 12, 16
with ISPF 109
with SQL/DS 111
VS FORTRAN 25, 49
brief description 112, 123
discussion 50
testing 58
using 25
with ISPF 109
with SQL/DS 111
VSAM
See VSE/VSAM
VSE/VSAM
brief description 112
definition 122
documenting tool 93
VTAM
See also ACF/VTAM
definition 122

W

warning messages 40
windowing 38, 40
writing the publications
administration documentation 84
end-user documentation 83
Memo to Users 87
WRTERM macro 26

X

XEDIT 28, 53, 87
XEDIT for documenting 92

Y

your verification EXEC guidelines 74

Numerics

0 (zero) return code 72
111 return code 73
3270 PC File Transfer
  brief description 113
  definition 123
  documenting tool 93
3270 PC with 3270 PC File Transfer 113
777 return code 72
888 return code 73
999 return code 73

VM/Integrated System
Developing an Application: Getting Started
Order No. SC24-5305-01

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

_____

_____

_____

_____

IBM Branch Office serving you: _____

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

SC24-5305-01

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

SC24-5305-01

Printed in U.S.A.