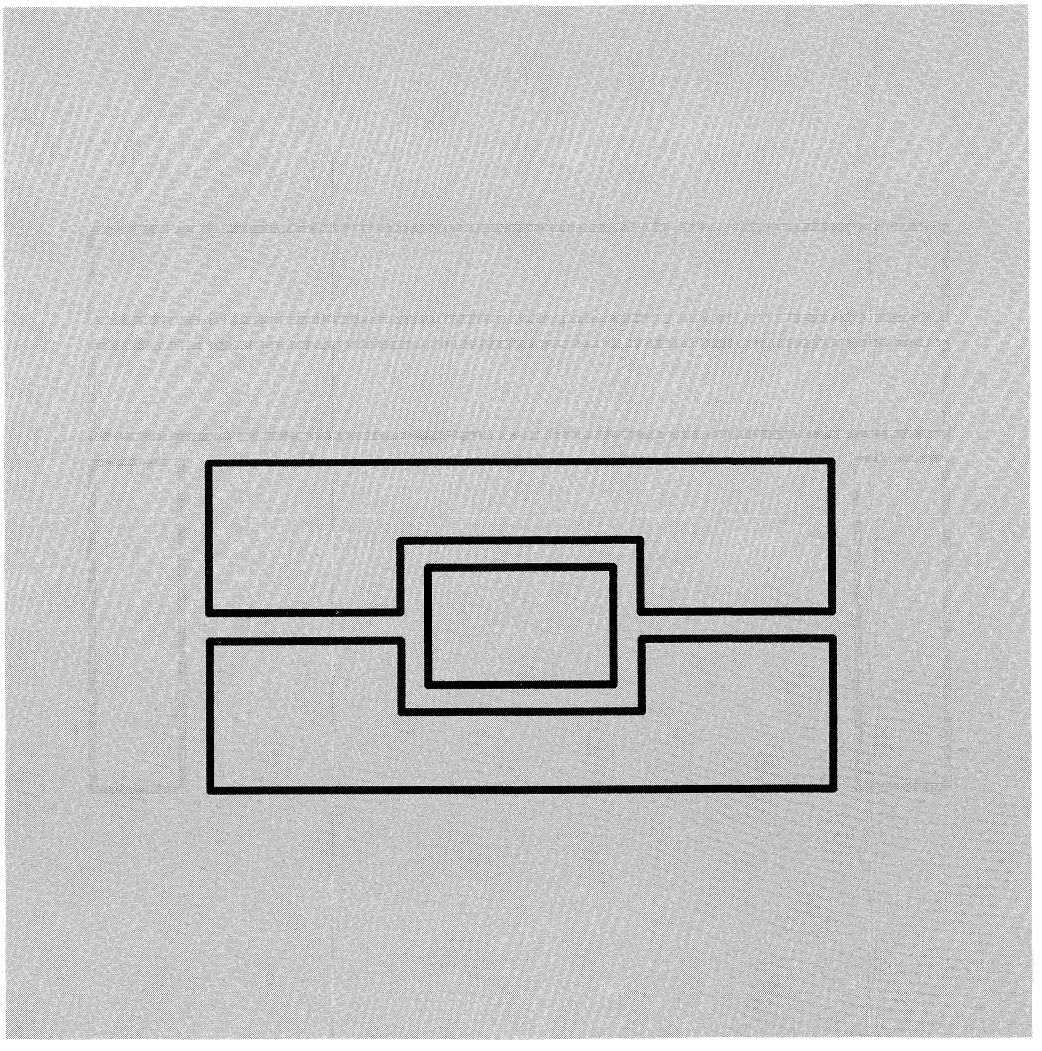
Systems Application Architecture

S544-3698-00

Common Programming Interface PrintManager Reference



Systems Application Architecture

S544-3698-00

Common Programming Interface PrintManager Reference

Note!

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

First Edition (May 1991)

This edition applies to IBM's Systems Application Architecture and to the following products:

Release 1 of IBM SAA PrintManager (5688-179),
Version 2 Release 1 of Operating System/400 (5738-SS1),

and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Be sure to use the correct edition for the level of the product.

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

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, Colorado 80301-9191

Contents

Figures

Tables

Notices

References in this publication to products or services of IBM do not imply that IBM will make them available in all countries where IBM does business or that only products or services of IBM may be used. Noninfringing equivalents may be substituted, but the user must verify that such substitutes, unless expressly designated by IBM, work correctly. No license, expressed or implied, to patents or copyrights of IBM is granted by furnishing this document.

Programming Interface

This [*publication*] is intended [*to help the customer to do*] [*programming with the PrintManager Interface*]. This [*publication*] documents General-Use Programming Interface and Associated Guidance Information provided by [*PrintManager*].

General-Use programming interfaces allow the customer to write programs that obtain the services of [*PrintManager*].

Trademarks

The following are trademarks of the IBM Corporation:

Advanced Function Printing	AFP
Application System/400®	AS/400®
Graphical Data Display Manager	GDDM
IBM®	
IBM 3800 Page Printing Subsystem	
IBM 3812 Page Printer	
IBM 3816 Page Printer	
IBM 3820 Page Printer	
IBM 3825 Page Printer	
IBM 3827 Page Printer	
IBM 3835 Page Printer	
IBM SAA PrintManager	
MVS	
MVS/ESA	
Operating System/2®	OS/2®
Operating System/400®	OS/400®
Print Services Facility	PSF
Systems Application Architecture	SAA
Virtual Machine/Extended Architecture	VM/XA
VM	

The following are trademarks of other companies:

PostScript® Adobe Systems Incorporated

Chapter 1. Introduction

This chapter:

- Identifies the book's purpose and audience
- Gives an overview of PrintManager, its components, and their relationship
- Gives a brief overview of the Systems Application Architecture (SAA) solution
- Explains how to use the book
- Describes related publications.

Who Should Read This Book

This book defines the SAA PrintManager interface. It is intended for programmers who want to write applications that adhere to this definition.

This book is a reference rather than a tutorial. It assumes you are already familiar with basic programming concepts.

What PrintManager Is

PrintManager is the collective name of a group of IBM licensed programs or operating system functions designed to provide common access to printing, including Advanced Function Printing (AFP), across the supported environments. PrintManager consists of the following:

- The product implementation of the PrintManager Interface, an element of the IBM Systems Application Architecture (SAA) common programming interface (CPI). This implementation allows you to write portable applications for sending print files to a system spool for printing.

  Among other benefits, the PrintManager Interface allows you to specify and validate print-option values (such as those currently specified with MVS SYSOUT JCL parameters or with the PSF/VM PSF command) from within an application in a form that is consistent across the supported environments. Applications that use the PrintManager Interface, therefore, are portable because they can be developed for one environment and used with little or no modification in another environment.

- An Application Programming Interface (API) that allows you to create print descriptors that can contain common information about printer routing, printer capabilities, and printer and job defaults. Applications that use the PrintManager Interface can use print descriptors created by the API.

  To create and maintain print descriptors, you can use either:

  - For the VM and MVS environments, the Prd Tool (PrdT), which includes a set of tags and commands, or
  - The API verbs.

- For the VM and MVS environments, a Print Request Facility (PRF) that provides the casual user with a consistent way to submit print jobs. The PRF provides both a command interface and an interactive menu, which is Interactive System

Productivity Facility (ISPF) based. You can customize the interactive menu (and its functions) to meet the needs of your organization.

The PrintManager components provide capabilities to handle major printing functions within each environment, as shown in Table 1-1.

Table 1-1. PrintManager Components by Environments

PrintManager Component	MVS	VM	OS/400	OS/2
PrintManager Interface	X	X	X	
Prd Tool (PrdT)	X	X		
Application Programming Interface (API) Verbs	X	X	X	
Print Request Facility (PRF)	X	X		

PrintManager is packaged as follows:

- For MVS and VM customers, as the IBM SAA PrintManager licensed program, which includes the PrintManager Interface, Prd Tool, API verb, and PRF components.
- For OS/400 customers, the PrintManager Interface and API verb components are included within the operating system and are known as PrintManager/400.

Note: PrintManager provides C programming language support for the PrintManager Interface and API verbs in the OS/400, MVS, and VM environments. PrintManager also provides RPG and COBOL programming language support for the PrintManager Interface verbs in the OS/400 environment.

PrintManager provides print-management capability within a single environment and enhances the ability to route print jobs from one system to another, using a communication program. For example, you can send jobs from one IBM System/370 or IBM System/390 to another, using the existing networking facilities (Remote Spooling Communication Subsystem [RSCS] for VM and Job Entry Subsystem [JES2] for MVS).

PrintManager defines a set of print options that are consistent across the supported environments, and it allows you to specify these print options within the application. PrintManager also provides the ability to:

- Select AFP resources from a system library on the printing system
- Package AFP resources with the print job (*inline*) when you send the job from one system to another. Refer to your printer-driver documentation for information on how inline resources are handled in each environment.

These print options, combined with the ability to create common, portable applications and printing definitions (with print descriptors), make possible easy and consistent access to printing across your organization. The PRF component of IBM SAA PrintManager provides additional ease of use and function for submitting print jobs.

To summarize, with PrintManager you can:

- Use the PrintManager Interface and API verbs (or, for MVS and VM, the PrdT) to create batch applications or installation-specific end-user interfaces for printing

- Use the PRF to submit print jobs.

What the SAA Solution Is

The SAA solution is based on a set of software interfaces, conventions, and protocols that provide a framework for designing and developing applications.

The SAA solution:

- Defines a common programming interface that you can use to develop consistent, integrated enterprise software
- Defines common communication support that you can use to connect applications, systems, networks, and devices
- Defines a common user access that you can use to achieve consistency in screen layout and user-interaction techniques
- Offers some applications and application development tools written by IBM.

Supported Environments

Several combinations of IBM hardware and software have been selected as SAA environments. These are environments in which IBM will manage the availability of support for all applicable SAA elements and the conformance of those elements to SAA specifications. The SAA environments are the following:

- MVS
  - Base system (TSO/E, APPC/MVS, batch)
  - CICS
  - IMS
- VM with CMS
- Operating System/400 (OS/400)
- Operating System/2 (OS/2).

Common Programming Interface

As its name implies, the common programming interface (CPI) provides languages and services that programmers can use to develop applications which take advantage of SAA consistency. These applications can be easily integrated and transported across the supported environments.

The components of the CPI fall into two general categories:

- Languages

  Application Generator
  C
  COBOL
  FORTRAN
  PL/1
  Procedures Language
  RPG.

- Services

  Communications Interface
  Database Interface

Dialog Interface
Presentation Interface
PrintManager Interface
Query Interface
Repository Interface.
Resource Recovery Interface.

The CPI is defined by this and the other CPI reference books. The CPI is not in itself a product or a piece of code. But—as a definition—it does establish and control how IBM products are being implemented, and it establishes a common base across the applicable SAA environments.

Thus, when you want to create an application that can be used in more than one environment, you can stay within the boundaries of the CPI and obtain easier portability. (Naturally, the design of such applications should be done with portability in mind as well.)

A list of SAA books to help you can be found under "Related Documentation" on page 1-5 and on the back cover of this book.

How to Use This Book

This book is organized as follows:

- Chapter 1, "Introduction"—Identifies the purpose and audience of the book, gives a brief overview of SAA, and explains how to use the book.
- Chapter 2, "Overview of the PrintManager Interface"—Provides an overview of the PrintManager Interface.
- Chapter 3, "Using the PrintManager Interface"—Describes a PrintManager Interface session, gives the calling sequence for the PrintManager Interface verbs, and tells how to diagnose errors.
- Chapter 4, "PrintManager Interface Verb Reference"—Contains reference information for the PrintManager Interface.
- Chapter 5, "Print Options"—Provides information on defaults and validation rules for print options, and lists print options.
- Appendix A, "PrintManager Print Options"—Provides detailed information of PrintManager print options.
- Appendix B, "PrintManager C Applications"—Provides verb bindings for C language, and an example of a C programming language example.
- Appendix D, "PrintManager RPG Applications"—Provides verb bindings for RPG language, and an example of a RPG programming language example.
- Appendix C, "PrintManager COBOL Applications"—Provides verb bindings for COBOL language, and an example of a COBOL programming language example.
- Appendix E, "System-Specific Information"—Describes unique implementation of the PrintManager Interface in the VM, MVS, and OS/400 operating-system environments.
- Appendix F, "Verb Error Codes"—Describes verb error codes.
- Appendix G, "IBM SAA PrintManager Operating Environment"—Lists the hardware and software prerequisites for IBM SAA PrintManager.

- Glossary—Defines technical terms.

Relationship to Products

The PrintManager Interface defines the elements that are consistent across the applicable SAA environments. Preparing and running programs requires the use of the PrintManager product that implements the interface on one of these systems.

For the PrintManager Interface, these products are:

- IBM SAA PrintManager for MVS and VM
- The PrintManager functions in the OS/400 operating system (known as PrintManager/400).

How Product Implementations Are Designated

Because the SAA solution is still evolving, complete and consistent products may not be available yet on all the applicable systems. Some interface elements may not be implemented everywhere. Others may be implemented, but differ slightly in their syntax or semantics (how they are coded and how they behave at run time).

These conditions are identified in this book in two ways:

- A system checklist precedes each interface element. If the interface element is implemented or announced on a particular system, that column is marked with an X. If it is not, that column is blank. In the following example, an element is implemented or announced on MVS, VM, and OS/400; but not on OS/2.

Table 1-2. Sample Verb System Checklist

MVS	VM	OS/400	OS/2
X	X	X	

- The PrintManager interface definition is printed in black ink. ***If the implementation of an interface element in an operating environment differs from the SAA definition in its syntax or semantics, text is printed in green— as in this sentence.***

Related Documentation

For more information about SAA, PrintManager, OS/400, VM, MVS, the supported languages, and printing (including AFP), refer to the appropriate publications listed in this section.

For the SAA Solution

An introduction to the SAA solution in general can be found in *SAA Overview*, GC26-4341.

An introduction to the common programming interface can be found in *Common Programming Interface: Summary*, GC26-4675.

More detailed information on the components of the common programming interface is available in the following SAA manuals (including this one):

Application Generator Reference, SC26-4355
C Reference—Level 2, SC09-1308
COBOL Reference, SC26-4354

Communications Reference, SC26-4399
Database Reference, SC26-4348
Dialog Reference, SC26-4356
FORTRAN Reference, SC26-4357
PL/I Reference, SC26-4381
Presentation Reference, SC26-4359
PrintManager Reference, S544-3698
Procedures Language Reference, SC26-4358
Procedures Language Level 2 Reference, SC24-5549
Query Reference, SC26-4349
Repository Reference, SC26-4684
RPG Reference, SC09-1286.

General programming advice can be found in *Writing Applications: A Design Guide*, SC26-4362. An introduction to the use of the AD/Cycle application development tools can be found in *AD/Cycle Concepts*, GC26-4531.

A definition of the common user access can be found in *Common User Access: Advanced Interface Design Guide*, SC26-4582, and *Common User Access: Basic Interface Design Guide*, SC26-4583.

General programming advice may be found in *Writing Applications: A Design Guide*, SC26-4362. An introduction to the use of the AD/Cycle application development tools can be found in *AD/Cycle Concepts*, GC26-4531.

A definition of the common user access can be found in *Common User Access: Advanced Interface Design Guide*, SC26-4582, and *Common User Access: Basic Interface Design Guide*, SC26-4583.

More information on the common communications support can be found in *Common Communications Support: Summary*, GC31-6810.

An introduction to distributed data in the SAA world can be found in *Concepts of Distributed Data*, SC26-4417.

More information on SAA system management can be found in *An Introduction to SystemView*, GC23-0576.

Ordering Information: Contact your local IBM branch office for information on how to order the above publications. They also can be obtained through an authorized IBM dealer.

The entire set of SAA publications can be ordered by specifying the bill-of-forms number SBOF-1240.

For PrintManager

You should use this book with three companion publications in the PrintManager library:

IBM SAA PrintManager Overview, S544-3347, which provides general information about the IBM SAA PrintManager licensed program. It further describes the three IBM SAA PrintManager components (the API, the PrintManager Interface, and the PRF), discusses the benefits of IBM SAA PrintManager, shows how to use IBM SAA PrintManager for distributed printing, and gives examples of how to use IBM SAA PrintManager in some typical business environments.

PrintManager Application Programming Interface Reference, S544-3699, which provides reference and use information about the API in its supported environments.

IBM SAA PrintManager User's Guide, S544-3346, which provides information about the IBM SAA PrintManager PRF.

For OS/400

For more information about OS/400 and the PrintManager functions within OS/400, refer to:

Application System/400 Information Directory, GC41-9678, for a description of the information in the Application System/400 (AS/400) library.

Application System/400 System Introduction, GC41-9766, for an overview of the features and capabilities of the AS/400 system.

Application System/400 Guide to Programming for Printing, SC41-8194, for information on the printing functions in AS/400.

Application System/400 Programming: CL Reference, SBOF-0481, for reference information on the CL commands.

For VM

For more information about using PrintManager in the VM operating system, refer to:

Virtual Machine/System Product CMS User's Guide, SC19-6210, *IBM Virtual Machine/Extended Architecture CMS User's Guide*, SC23-0356, or *IBM Virtual Machine/Enterprise Systems Architecture CMS User's Guide*, SC24-5460 for information on using CMS commands.

Virtual Machine/System Product CMS Command Reference, SC19-6209, *IBM Virtual Machine/Extended Architecture CMS Command Reference*, SC23-0354, or *IBM Virtual Machine/Enterprise Systems Architecture CMS Command Reference*, SC24-5461, for reference information on CMS commands.

Virtual Machine/System Product CP Command Reference, SC19-6211, *IBM Virtual Machine/Extended Architecture CP Command Reference*, SC23-0358, or *IBM Virtual Machine/Enterprise Systems Architecture CP General User Command Reference*, SC24-5433 for reference information on CP commands.

For MVS

For more information on JCL commands, refer to *MVS/ESA JCL Reference*, GC28-1829.

For more information on MVS error codes, refer to *MVS/ESA System Programming Library: Application Development Guide*, GC28-1852, GC28-1854, and GC28-1857.

For C Language

For more information on writing C language applications, refer to:

IBM C/370 User's Guide, SC09-1264

Application System/400 Languages: C/400 User's Guide, SC09-1303.

For RPG Language

For more information on writing RPG language applications, refer to:

AS/400 Languages: RPG/400 User's Guide, SC09-1161

AS/400 Languages: RPG/400 Reference, SC09-1089.

For COBOL Language

For more information on writing Cobol language applications, refer to:

AS/400 Languages: COBOL/400 User's Guide, SC09-1158

AS/400 Languages: COBOL/400 Reference, SC09-1240.

For Printing

For more information about printing (including AFP), refer to the following publications:

A Guide to IBM's Advanced Function Printing, G544-3095, which contains an overview of AFP, its benefits, and the AFP licensed programs. This book also provides examples of typical AFP applications.

Advanced Function Printing: Software General Information, G544-3415, which describes IBM's AFP licensed programs and contains a list of the publications for the various AFP products.

Advanced Function Printing: Printer Summary, G544-3135, which lists the printers supported by AFP software.

Advanced Function Printing: Data Stream Reference, S544-3202, which defines the AFP data stream.

Print Services Facility/VM: Application Programming Guide, S544-3466 or *Print Services Facility/MVS: Application Programming Guide*, S544-3084, for information about PSF application programming.

Print Services Facility/VM: System Programming Guide, S544-3467 or *Print Services Facility/MVS: System Programming Guide*, SH35-0091, for information PSF system programming.

Application System/400 Guide to Programming for Printing, SC21-8194, for information on the printing functions in AS/400.

For GDDM

For information about GDDM device types, refer to *GDDM Base Programming Reference*, SC33-0332.

Interface Definition Table

The table below lists the elements currently in the PrintManager Interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program available that implements a particular element.

Table 1-3. Major Elements of the PrintManager Interface

PrintManager Interface Verb	MVS[1]	VM	OS/400	OS/2
SPRABRT (Abort)	X	X	X	
SPRADOC (Abort Document)	X	X	X	
SPRADDF (Add File)	X	X	X	
SPRCLOS (Close)	X	X	X	
SPREDOC (End Document)	X	X	X	
SPRFERI (Free Error Information)	X	X	X	
SPRGEEM (Get Error Message)	X	X	X	
SPRGERI (Get Error Information)	X	X	X	
SPRLOPT (List Options)	X	X	X	
SPROPEN (Open)	X	X	X	
SPRQOPT (Query Option)	X	X	X	
SPRSDOC (Start Document)	X	X	X	
SPRSOPT (Set Option)	X	X	X	
SPRWRIT (Write)	X	X	X	
Supported Languages				
C	X	X	X	
RPG			X	
COBOL			X	

note:

[1] Available to TSO/E and MVS batch job users only.

Chapter 2. Overview of the PrintManager Interface

Applications that print, normally go through a two-step process:

1. You can use an application that formats the output data and writes the data to a logical file.
2. You can use operating-system-specific controls (for example, JCL) to define print options and submit the print job.

The PrintManager Interface provides a one-step process to define print options as part of the application. This eliminates the need to use operating-system-specific controls. It provides access to print functions in all supported environments through a common set of services that can be called from the application. This helps you create consistent, easy-to-use print applications that can be sent to any supported environment. By writing to the PrintManager Interface, you obtain the following advantages:

Consistency in Printing: The PrintManager Interface provides you with a consistent way to submit print jobs in all supported environments. This includes:

- A method for print-job submission that is independent of any data stream or protocol
- A consistent definition of print options
- A common, system-independent way of specifying the destination of a print job.

Enhanced Application Portability: A print application developed with the PrintManager Interface can be migrated to any supported environment with little or no modification. For example, a print application developed for VM can be adapted to MVS with few changes.

Access to Advanced Function Printing: The PrintManager Interface provides you with a structured way to specify print options that is adapted to the requirements of IBM's Advanced-Function Printing.

Who Should Use the PrintManager Interface

The PrintManager Interface does not replace the current high-level language statements to format data and create a print job. The PrintManager Interface is meant to complement these interfaces and to provide a system-independent way of specifying print options and submitting the job to the print spool once the output data has been described.

For those applications, such as an end-user interface, that specify print-option values as part of the application, the PrintManager Interface will be invaluable. In this situation, you submit a print job without requiring a separate step to define print options.

Unlike batch-print submission methods, print-job submission under the PrintManager Interface provides you with the ability to specify print options and a print destination as the print job is created. This integrated process provides the following:

- The ability to develop application programs that have their own interactive end-user interface for printing.

- The ability to control print options at the application level. For example, an application program can be developed to enforce a standard set of print options across the enterprise: A memo must always be printed with the company logo; a print job longer than five pages must always be duplexed to save paper costs.
- The ability to retrieve and place Advanced Function Printing (AFP) resources inline with a print job. For example, any form definitions or overlays can be associated with the print job as it is shipped over an SAA network. See your printer-driver documentation for information on how inline resources are handled in each environment.
- The ability to dynamically control and specify print options for multiple print jobs within an application. For example, you could specify that the first print job be printed with 200 copies with one overlay and that the second print job be printed with 50 copies with a different overlay.
- The ability to write buffers of data and entire data files within the same print job (existing high-level languages allow an application to write only a single buffer at a time).

How the PrintManager Interface Is Implemented

The PrintManager Interface defines a set of functions, known as verbs, to print documents easily and consistently across supported environments. The major elements are the PrintManager Interface verbs and the PrintManager print options. The API works closely with the PrintManager Interface. Print descriptors are created and maintained with the API and can be used by the PrintManager Interface. See Chapter 1, "Introduction" on page 1-1 for more information.

PrintManager Interface Verbs

The PrintManager Interface verbs are designed to be invoked from application programs. These verbs allow you to perform the following functions:

- Open a print session
- Define print options or modify print options
- Create a print job
- Abort a print job
- Retrieve error information
- Close a print session.

The following are the PrintManager Interface verbs:

SPRABRT (Abort): Aborts the print session.

SPRADOC (Abort Document): Aborts and deletes the print job.

SPRADDF (Add File): Writes a file of print data to the print job.

SPRCLOS (Close): Ends the current print session.

SPREDOC (End Document): Ends the print job.

SPRFERI (Free Error Information): Frees storage for the error-information structure returned from the SPRGERI (Get Error Information) verb.

SPRGEEM (Get Error Message): Returns additional error information associated with the error-information structure.

SPRGERI (Get Error Information): Returns the error-information structure from the last error.

SPRLOPT (List Options): Lists the print options currently defined.

SPROPEN (Open): Opens and identifies a print session.

SPRQOPT (Query Option): Lists the current value for a print option.

SPRSDOC (Start Document): Starts a print job and specifies a document name for the print job.

SPRSOPT (Set Option): Allows you to set or change a print option.

SPRWRIT (Write): Sends a buffer of data to the print job.

Chapter 3. Using the PrintManager Interface

This chapter does the following:

- Describes a PrintManager Interface session, including how to open a session, set print options, create a print job, and close the session
- Discusses briefly how to use print descriptors with the PrintManager Interface
- Describes the operating states that determine the calling sequence for the PrintManager Interface verbs
- Tells how to diagnose problems.

Note: For a C language programming example, see "Example of a PrintManager Session in C Language" on page B-31. For a COBOL language programming example, see "Example of a PrintManager Session in COBOL Language" on page C-31. For an RPG language programming example, see "Example of a PrintManager Session in RPG Language" on page D-34.

Using PrintManager Interface Verbs and Writing Applications

The PrintManager Interface allows an application program to place print jobs on the spool by providing a set of functions called verbs. You call the PrintManager Interface verbs from within an application program. Before you can use the PrintManager verbs, you must initialize PrintManager using the SPRINIT (Initialize PrintManager) verb. (See Appendix E, System-Specific Information for more information). To use any of the PrintManager Interface verbs, you must issue the SPROPEN (Open) verb before sending print jobs to the spool. Multiple print sessions can be active at any time within a single application program by using multiple SPROPEN (Open) verbs.[1] For each session, one or more print jobs can be created and sent to the spool, with different print options defined for each job. Each session created using SPROPEN (Open) is identified by a unique print session identifier. A print session is ended by using the SPRCLOS (Close) verb. After you are through using the PrintManager Interface verbs, you must terminate PrintManager using the SPRTERM (Terminate PrintManager) verb. (See Appendix E, System-Specific Information for more information).

Once PrintManager has been initialized using the SPRINIT (Initialize PrintManager) verb, the minimum set of PrintManager Interface verbs required to place print jobs on the spool are the SPROPEN (Open) verb, the SPRSDOC (Start Document) verb, the SPRWRIT (Write) verb or the SPRADDF (Add File) verb, the SPREDOC (End Document) verb, and the SPRCLOS (Close) verb. After you are through using the PrintManager Interface verbs, you must terminate PrintManager using the SPRTERM (Terminate PrintManager) verb.

[1] The VM environment is an exception; this system allows only one session to be open at a time.

The following table shows the major steps contained within a print session and lists the PrintManager Interface verbs associated with each step.

Steps	PrintManager Interface Verbs
Open session	SPROPEN (Open)
Manage print options	SPRSOPT (Set Option) SPRLOPT (List Options) SPRQOPT (Query Option)
Create a print job	SPRSDOC (Start Document) SPRWRIT (Write) SPRADDF (Add File) SPREDOC (End Document)
Close the session	SPRCLOS (Close)
Note: Several PrintManager Interface verbs are associated with more than one step and can be issued at different times during a session. For detailed information about the calling sequence for PrintManager Interface verbs, refer to "PrintManager Interface Operating States" on page 3-5.	

The following list describes the major steps contained within a print session and tells briefly how to use the associated verbs with each step.

1. Opening the Session

   The application program must open a print session before any communication with the spool can take place. To do this, use the SPROPEN (Open) verb to start the session. When the SPROPEN (Open) verb is called, the application program may define the set of current options by specifying a previously defined print descriptor. See "Using Print Descriptors with the PrintManager Interface" on page 3-3 for more information on using print descriptors.

   Note: After the session is opened, you can specify print-option values and print descriptors with the SPRSOPT (Set Option) verb.

2. Managing Print Options

   The next step in a print session allows the application program to define print options or to modify the set of current options. The set of current options is defined as the print options associated with the current print job. It contains a combination of the default values inherited from the print descriptor and the print-option values overridden by calls to SPRSOPT (Set Option).

   To set print options, the SPRSOPT (Set Option) verb can be called at any time after the SPROPEN (Open) verb is issued and between print jobs. When the SPRSOPT (Set Option) verb is called, the application program can do the following:

   - Define or modify print-option values in the set of current options. For example, the application program might specify that 15 copies are printed of the first print job and then override this print option so that only 10 copies are printed of the second print job.
   - Specify a new print descriptor that contains new option default values and new validation rules. See "Using Print Descriptors with the PrintManager Interface" on page 3-3 for more information on using print descriptors.

   As you can see, the SPRSOPT (Set Option) verb is particularly useful when processing multiple print jobs with different print options. If necessary, the

application program can use the SPRSOPT (Set Option) verb to define print options on a *per job basis*.

All print options are verified against a set of validation rules provided in the print descriptor (if one is specified). For more information, refer to Chapter 5, "Print Options" on page 5-1.

Note: The last valid print-option value specified overrides previous values specified for that option or the default specified in the print descriptor.

The SPRLOPT (List Options) verb and the SPRQOPT (Query Option) verb can be called anytime after the SPROPEN (Open) verb is issued. The application program can use the SPRLOPT (List Options) verb to provide a list of the set of current options and can use the SPRQOPT (Query Option) verb to query the value of an option in the list.

3. Creating the Print Job

   The next step in a print session allows the application program to create a print job and to send it to the spool for printing. The application program *must* issue the SPRSDOC (Start Document) verb to initiate a print job. A print session can contain multiple print jobs, and each print job is identified by a SPRSDOC (Start Document) - SPREDOC (End Document) pair.

   When SPRSDOC (Start Document) is issued, a spool file is opened, and the resource options are processed. The SPRSDOC (Start Document) verb can be used to assign the print job a document name.

   After a new print job has been started, data is written to the spool, using either the SPRWRIT (Write) verb or the SPRADDF (Add File) verb. The SPRWRIT (Write) verb writes a single buffer of data; the SPRADDF (Add File) verb writes a complete file of data. The SPRWRIT (Write) and SPRADDF (Add File) verbs can be intermixed within the same print job.

   The SPREDOC (End Document) verb ends the print job, and it is required before the print session can be closed.

4. Closing the Session

   Use the SPRCLOS (Close) verb to end the print session.

Using Print Descriptors with the PrintManager Interface

A print descriptor contains a set of default print-option values and any validation rules that are associated with those print options. The advantages of using a print descriptor are that:

- The standard information associated with each print job can be shared across an installation and does not have to be respecified each time the job is placed on the spool. This simplifies printing for you. For instance, you can select a print descriptor called PRT25 and access a predefined set of default values and options that allow the job to be sent to Printer 025. The routing information and default print options for Printer 025 are stored in the print descriptor; therefore, you do not need to be aware of this information and you do not need to respecify them each time.
- Validation rules for print options are defined in the print descriptor, so print options are checked for validity before they are placed on the spool.

Specifying a print descriptor is an optional part of opening a print session. If no print descriptor is specified, the set of current options is empty until print options are specified using the SPRSOPT (Set Option) verb. If the print descriptor is

specified, it is retrieved from storage, and the default values within the print descriptor are applied to the set of current options.

You may also specify a new print descriptor using the SPRSOPT (Set Option) verb. You may want to do this to specify new option default values and new validation rules. For example, the application program may specify that the option default values for the PRT25 print descriptor are associated with the first print job and that the option default values for the PRT014 print descriptor are associated with the second print job.

The print descriptors you use in your organization use the name formats shown in Figure 3-1.

```
Universal: PRDNAME=saaname
Relative:  GRPALIAS=pdgrp_alias PRDNAME=saaname
Exact:     GRPEXACT=sys_spec_prdgrp_name PRDNAME=saaname
```

Figure 3-1. Print-Descriptor Name Formats

PrintManager treats each name format as follows:

- A *universal* name refers only to a print descriptor.
- A *relative* name refers to both a print descriptor and its print-descriptor group via an *alias* name for the group.
- The *exact-name* format is similar to the relative-name format, except that system-specific names are used instead of group alias names.

Note: For more information on print descriptors, refer to the *PrintManager Application Programming Interface Reference* (S544-3699).

PrintManager Interface Operating States

The calling sequence of the PrintManager Interface verbs is determined by the internal operating states of the PrintManager Interface. These operating states are as follows:

Session Active

The Session Active state allows you to set, to modify, to list, or to query print options. You must issue the SPROPEN (Open) verb to go to the Session Active state. You must issue the SPRSDOC (Start Document) verb to go from Session Active state to Job-in-Progress state. Use the SPRCLOS (Close) verb or the SPRABRT (Abort) verb to close a session when in the Session Active state. The following verbs are valid in Session Active state:

SPRSOPT (Set Option)
SPRLOPT (List Options)
SPRQOPT (Query Option)
SPRSDOC (Start Document)
SPRABRT (Abort)
SPRCLOS (Close)
SPRGERI (Get Error Information)
SPRFERI (Free Error Information)
SPRGEEM (Get Error Message)

Job-in-Progress

The Job-in-Progress state indicates that the job has been started, and that data is being written to the spool. You must issue the SPREDOC (End Document) verb or the SPRADOC (Abort Document) verb to return to Session Active state. When in the Job-in-Progress state, you can use the SPRABRT (Abort) verb to abort the job and close the session. The following verbs are valid in the Job-in-Progress state:

SPRLOPT (List Options)
SPRQOPT (Query Option)
SPRWRIT (Write)
SPRADDF (Add File)
SPRADOC (Abort Document)
SPREDOC (End Document)
SPRABRT (Abort)
SPRGERI (Get Error Information)
SPRFERI (Free Error Information)
SPRGEEM (Get Error Message)

If you issue a PrintManager Interface verb in the wrong operating state, PrintManager returns an error.

Handling Problems

The following sections tell how to handle problems when using the PrintManager Interface, which consists of:

- Using the PrintManager error verbs to find the cause of errors that occur while using the other PrintManager Interface verbs.
- Using the PrintManager trace facility to gather information about problems with PrintManager (if requested by your IBM Support Center representative).

Using PrintManager Error Verbs

Except for the SPRINIT (Initialize PrintManager) and SPRTERM (Terminate PrintManager) verbs, if a PrintManager verb returns a false or 0 return value, you can use the PrintManager error verbs to obtain information about the error as follows:

- Use the SPRGERI (Get Error Information) verb to get the error information associated with the last PrintManager Interface error. The error information returned includes an error code and the severity of the error.

  A severity code of 4 indicates a warning condition, while a severity code of 8 indicates an error condition. A warning condition means that the requested function completed successfully, and you may continue, although results may not be as expected. For example, an error of X'40B1' (16561) has a severity code of 4, and indicates that truncation occurred when a valid value was set in a print descriptor. An error condition means that an error occurred, and the requested function did not complete successfully.

- For some errors, additional error information can be obtained by using **ERRINFO** (as returned from SPRGERI (Get Error Information)) as input to the SPRGEEM (Get Error Message) verb. The SPRGEEM (Get Error Message) verb returns error information that contains a message identifier and a value that contains additional error information.

Note: Appendix F, Verb Error Codes lists PrintManager error codes and provides general explanations of associated errors, a list of PrintManager Interface verbs that may have issued the error, user responses, and a list of additional messages (if any) that are issued.

- After **ERRINFO** is no longer needed, use the SPRFERI (Free Error Information) verb to free storage obtained for the error.

Using the Trace Facility

After you have used the error verbs, if you suspect a problem with PrintManager, you should contact your IBM Support Center representative, who may ask you to run a trace to gather more information. This section tells how to:

- Create a trace setup file
- Start a trace
- Stop a trace.

Creating a Trace Setup File

You specify trace options in a *trace setup file* that you create (see "Starting a Trace" on page 3-8).

Notes:

1. If you are using a file editor that can insert line numbers in a file, ensure that you do *not* create a trace setup file with line numbers. A file with line numbers cannot be used as a trace setup file.
2. Keywords must be in upper case or the SPRINIT (Initialize PrintManager) verb will fail.
3. Values must also be in upper case, or they are ignored.

In the trace setup file, you code trace options in the **KEYWORD = VALUE** format and in the order shown in the following list (default values are underscored):

PRTMGR_TRACE_FLAG
Specifies whether tracing is enabled.

TRUE
Tracing is enabled.

FALSE
Tracing is disabled.

PRTMGR_TRACE_FILE = *filename*
Filename specifies the name of the file where trace output will be sent. This keyword and file name is required to successfully complete a trace. In MVS, the output file name is a sequential data set name, in VM it is a file name, and in OS/400 it is a database file name.

If the output file does not exist it will be created. In MVS, the dataset is created with your high level qualifier. In VM, the file name and file type are required. In OS/400, the file name is required.

Note: Ensure that you have write access to the VM minidisk, MVS high level qualifer of the data set, or OS/400 library for the output file. Otherwise, a SPRINIT (Initialize PrintManager) error will occur.

PRTMGR_WRAP
Specifies whether the trace will wrap.

WRAP
Trace will wrap.

NOWRAP
Trace will not wrap (tracing stops when the trace output file is full).

PRTMGR_FILE_SIZE = *size*
Specifies the maximum size (in bytes) of the trace entries before wrapping occurs. If you specify a value of less than 32767, PrintManager will change the value to 32767, which is the default value for all system environments.

PRTMGR_TRACE_CLASS
Specifies the PrintManager verbs to be traced. This keyword and value is required and there is no default.

PRTMGR
Specifies tracing of all the PrintManager Interface verbs.

PRD
Specifies tracing of all the API verbs.

Starting a Trace

To start a trace, you must create a trace setup file (see "Creating a Trace Setup File" on page 3-7). The SPRINIT (Initialize PrintManager) verb will then read the trace file and start the trace. If no setup file exists or if PrintManager cannot reference the file, tracing will not occur. The following sections tell how to name and specify a trace setup file.

Starting MVS and VM Traces: On VM and MVS systems, the trace setup file must be associated with the ddname **EKITRACE**. In MVS, you can use JCL or the TSO ALLOCATE command to associate a data set name with this ddname. In VM, you can use the CMS FILEDEF command.

Starting OS/400 Traces: On OS/400 systems, the trace setup file must be specified via a database file with the general name **EKITRACE**. To actually start the trace, however, it is recommended that you override the file name **EKITRACE** with an actual database file name with the OVRDBF command. If the override file does not exist, it will be created with a record length of 200 bytes. If you create the override file, it is recommended that the override file have a record length of 200 for a complete trace. If you do not override the file name, then the **EKITRACE** data base file must exist in a library in your library list.

Stopping a Trace

The following sections describe the system-specific methods you use to stop a trace.

Stopping an MVS Trace: On MVS systems, you can stop a trace by either removing the **EKITRACE** ddname or by using the TSO FREE command.

Stopping a VM Trace: On VM systems, you stop a trace with the CMS FILEDEF CLEAR command.

Stopping an OS/400 Trace: On OS/400 systems, you can stop a trace with the DLTOVR command which deletes the override of the file name **EKITRACE**. If you do not override the file name, then delete the **EKITRACE** file or remove it from your library list.

Creating an Echo File

Your IBM Support Center representative may ask you to produce an echo file. PrintManager provides a special option, ECHO, that can be set using the SPRSOPT (Set Option) verb. The value of the ECHO option is a string containing the name of a file that will receive the echo information. The file name follows the system-specific naming convention for each environment:

VM filename filetype [filemode] or filename.filetype[.filemode]

MVS QUAL1.QUAL2.QUAL3.QUAL4.QUAL5

OS/400 library/file(member)

Note: In OS/400, the file name specified must be a database file. If the file does not exist, it will be created with a record length of 200 bytes.

Chapter 4. PrintManager Interface Verb Reference

This chapter contains reference information for the PrintManager Interface verbs and consists of the following sections:

- "Format of the PrintManager Interface Verb Descriptions," which shows the format used to describe the PrintManager Interface verbs.
- "PrintManager Interface Verb Data Types" on page 4-2, which describes the general data types and data structures used by the PrintManager Interface verbs. These data types and structures are shown in a *metalanguage* that corresponds to the verb format used in this chapter.
- The descriptions of each PrintManager Interface verb.

To supplement the descriptions of the verbs provided in this chapter, see the following:

- For information on using the PrintManager Interface and its verbs, see Chapter 3, Using the PrintManager Interface.
- For C language programming information, see Appendix B, "PrintManager C Applications" on page B-1.
- For COBOL language programming information, see Appendix C, "PrintManager COBOL Applications" on page C-1.
- For RPG language programming information, see Appendix D, "PrintManager RPG Applications" on page D-1.
- For verb error data, see Appendix F, Verb Error Codes.
- For information on PrintManager print options, see Chapter 5, Print Options and Appendix A, PrintManager Print Options.

Format of the PrintManager Interface Verb Descriptions

Each PrintManager Interface verb description contains these sections:

Verb name — The PrintManager programming verb name and the natural language verb name (in parentheses).

Supported system checklist — The operating systems supported by the verb.

Function — A brief description of the verb's function.

Syntax — The verb syntax, which is shown in a *metalanguage* using the PrintManager programming verb name. For information on the actual verb calls for a specific language, see the language appendixes.

Parameters — The parameters and their general data types. The parameter description also tells whether the parameter is an input, output, or input/output parameter and tells how to use the parameter.

Usage — Programming use information for the verb.

Return values — The name of the return parameter of the verb and its return values, or TRUE/FALSE indicators if a Boolean function.

Environment restrictions

Operating-system-specific information for the verb.

PrintManager Interface Verb Data Types

This section describes the general data types for the PrintManager Interface verbs described in this chapter:

Data Type	Description
BOOL	An integer with zero or nonzero values
BUFFER	An array or structure
BYTE	A byte of data
CHAR	A single-byte character
ERRINFO	The SPRGERI (Get Error Information) verb returns error information that consists of the error ID and additional error messages that can be obtained by using the SPRGEEM (Get Error Message) verb.
ERRMSG	Additional error information returned by the SPRGEEM (Get Error Message) verb.
ERRORID	Provides the severity and error code for an error condition
HAB	A unique identifier (handle) for a PrintManager anchor block
HPRM	A print session identifier
LONG	A 32-bit integer
NULL	A field with a zero value.
OPTDATA1	The SPRSOPT (Set Option) and SPRQOPT (Query Option) verbs use an OPTDATA1 that consists of an option, value type, value length, and a value.
SDF	The SPRLOPT (List Options) verb uses a self defining field.
STRL	A string with an implicit length count
STRLARR	An array of character strings with implied lengths.
ULONG	An unsigned 4-byte integer value (32 bits)
USHORT	An unsigned 2-byte integer value (16 bits)

SPRABRT (Abort)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Aborts the print session.

Syntax

```
SPRABRT (hprm, success)
```

Figure 4-1. Format of the SPRABRT (Abort) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Usage

Use the SPRABRT (Abort) verb to abort the print session and to delete the print job (if active). When the SPRABRT (Abort) verb is issued, the *hprm* from the aborted session is no longer valid.

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Ending your application without calling SPRCLOS (Close) or SPRABRT (Abort) could cause unpredictable results.

Return Values

***success* (BOOL)**
A nonzero value means the print session was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADOC (Abort Document)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Aborts and deletes the print job.

Syntax

```
SPRADOC (hprm, success)
```

Figure 4-2. Format of the SPRADOC (Abort Document) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Usage

Use the SPRADOC (Abort Document) verb to abort processing of the print job and to delete it. When the SPRADOC (Abort Document) verb is issued, PrintManager returns to Session Active state.

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

SPRADOC (Abort Document) does not abort a print job that has already been ended with the SPREDOC (End Document) verb.

Return Values

***success* (BOOL)**
A nonzero value means the print job was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADDF (Add File)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Writes a file of print data to the print job.

Syntax

```
SPRADDF (hprm, FileName, success)
```

Figure 4-3. Format of the SPRADDF (Add File) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

FileName (STRL)
(Input). A string containing the name of a file with print data to be sent to the spool. The file name follows the system-specific naming convention for each environment:

VM filename filetype [filemode] or filename.filetype[.filemode]

MVS QUAL1.QUAL2.QUAL3.QUAL4.QUAL5

OS/400 library/file(member)

Usage

Use the SPRADDF (Add File) verb to write a file of print data to the print job.

If an error occurs, PrintManager returns to Session Active state with the exception of two errors: 16412 ("cannot open file") and 16547 ("read error"). These two errors will leave you in Job-in-Progress state. See "PrintManager Interface Operating States" on page 3-5 for more information on operating states.

Note: In OS/400, the file specified in the *FileName* parameter must be a physical data file with FILETYPE (*DATA) and LVLCHK (*NO).

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**

A nonzero value means the named file was written to the print job, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRCLOS (Close)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Ends the current print session.

Syntax

```
SPRCLOS (hprm, success)
```

Figure 4-4. Format of the SPRCLOS (Close) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Usage

Use the SPRCLOS (Close) verb to end a print session and to free storage associated with the session. When SPRCLOS (Close) is issued, the *hprm* parameter from the closed session is no longer valid.

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Ending your application without calling SPRCLOS (Close) or SPRABRT (Abort) could cause unpredictable results.

Return Values

***success* (BOOL)**
A nonzero value means the print session ended successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPREDOC (End Document)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Ends the print job.

Syntax

```
SPREDOC (hprm, success)
```

Figure 4-5. Format of the SPREDOC (End Document) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Usage

Use SPREDOC (End Document) to end a print job. When the SPREDOC (End Document) verb is issued, PrintManager returns to Session Active state.

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (USHORT)**
A nonzero value means the print job ended successfully, whereas a value of 0 means an error occurred. ***In VM, the non-zero return value is the VM print job ID.***

Environment Restrictions

None.

SPRFERI (Free Error Information)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Frees storage for the error information returned from the SPRGERI (Get Error Information) verb.

Syntax

```
SPRFERI (ErrorInfo, success)
```

Figure 4-6. Format of the SPRFERI (Free Error Information) Verb

Parameters

ErrorInfo (ERRINFO)
(Input). The **ERRINFO** buffer.

Usage

Use SPRFERI (Free Error Information) to free storage associated with the **ERRINFO** buffer. You should issue SPRFERI (Free Error Information) for each instance of the error information returned with the SPRGERI (Get Error Information) verb.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

***success* (BOOL)**
A nonzero value means storage was freed successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGEEM (Get Error Message)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Gets additional error information associated with the **ERRINFO**.

Syntax

```
SPRGEEM (ErrorInfo, Index, ErrorMsg, TotalCount, success)
```

Figure 4-7. Format of the SPRGEEM (Get Error Message) Verb

Parameters

ErrorInfo (ERRINFO)
(Input). The **ERRINFO** returned by SPRGERI (Get Error Information).

Index (ULONG)
(Input). Specifies which additional message to return. Use a value of 1 to select the first message, a value of 2 to select the second message, and so on.

ErrorMsg (ERRMSG)
(Output). The buffer containing **ERRMSG**.

TotalCount (ULONG)
(Output). SPRGEEM (Get Error Message) updates this field to specify the number of additional messages associated with the **ERRINFO**.

Usage

The SPRGEEM (Get Error Message) verb returns additional error information in **ERRMSG**. The original **ERRINFO** returned from SPRGERI (Get Error Information) must be passed to SPRGEEM (Get Error Message). It must not be a copy.

Information in **ERRMSG** is often useful in further resolving the cause of errors. For example, for error code 16412 ("cannot open file"), the *value* field of **ERRMSG** provides the name of the file where the error occurred. For operating system errors, PrintManager updates the *value* field with error text provided by the operating system. Information in the *value* field is truncated to 256 characters, if necessary.

There may be 1 to *n* additional error messages associated with the **ERRINFO**. Set *Index* to 0 to get the total number of messages that can be returned. If the value returned in *TotalCount* is 0, there are no additional messages. If there are additional messages, set *Index* to correspond to the message you want to return. If you want to return all additional messages, call SPRGEEM (Get Error Message) repeatedly, incrementing *Index* until *TotalCount* is reached.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

***success* (BOOL)**

A nonzero value means the requested function completed successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGERI (Get Error Information)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Gets the error information (**ERRINFO**) from the last error.

Syntax

```
SPRGERI (hab, ERRINFO)
```

Figure 4-8. Format of the SPRGERI (Get Error Information) Verb

Parameters

hab (HAB)
(Input). The anchor-block handle returned from a successful call to SPRINIT (Initialize PrintManager). See Appendix E, "System-Specific Information" on page E-1 for more information on SPRINIT (Initialize PrintManager).

Usage

Use the SPRGERI (Get Error Information) verb to get the **ERRINFO** associated with the last PrintManager Interface error. Associated with the error information is an error id. You must use the error information (**ERRINFO**) as input to the SPRGEEM (Get Error Message) verb to get additional information (if any) about the error. To free storage for the error information, use SPRFERI (Free Error Information).

For the** hab **parameter, ensure that you enter the anchor-block handle returned by the SPRINIT (Initialize PrintManager) verb. Otherwise, in the 370 environment PrintManager will abend with an abend code of X'0245' and in OS/400 a value of 0 will be returned.

Notes:

1. If PrintManager cannot be initialized (with SPRINIT (Initialize PrintManager)), SPRGERI (Get Error Information) will not return error information.
2. Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

***ERRINFO* (ERRINFO)**

The SPRGERI (Get Error Information) verb returns **ERRINFO** for the last error. See Appendix F, "Verb Error Codes" on page F-1 for information on PrintManager Interface verb error codes. A return value of 0 indicates that no previous error occurred.

As well as errors from PrintManager Interface verbs, you can get errors from API verbs because some PrintManager Interface verbs invoke PrintManager API verbs.

Environment Restrictions

None.

SPRLOPT (List Options)

MVS	VM	OS/400	OS/2
X	X	X	

Function

List current print options for a print session.

Syntax

```
SPRLOPT (hprm, Length, Buffer, ItemsReturned,
         ItemsRemaining, Continue, success)
```

Figure 4-9. Format of the SPRLOPT (List Options) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Length (ULONG)
(Input). Specifies the buffer size (see "Usage").

Buffer (BUFFER)
(Output). Returns a list of current print-option names. Each language has a specific format for this buffer.

ItemsReturned (ULONG)
(Output). Indicates the number of print-option names returned in the buffer.

ItemsRemaining (ULONG)
(Output). For a partial list, indicates the number of option names not yet listed; otherwise, *ItemsRemaining* is 0.

Continue (SDF)
(Input/Output). Used to get a list of options in a series of calls (see "Usage").

Usage

Use the SPRLOPT (List Options) verb to list the set of current options in a print session. The buffer size determines the number of options returned.

The *Continue* parameter can be set to specific language constants for a series of calls to SPRLOPT (List Options) to return print option names. For more information, refer to Appendix B, "PrintManager C Applications" on page B-1, Appendix C, "PrintManager COBOL Applications" on page C-1, and Appendix D, "PrintManager RPG Applications" on page D-1.

Note: Any changes to the print options during a series of calls using *Continue* will not be reflected in the partial lists returned.

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**

A nonzero value means the list request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPROPEN (Open)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Opens and identifies a print session.

Syntax

```
SPROPEN (hab, PrdName, Length, Buffer, hprm)
```

Figure 4-10. Format of the SPROPEN (Open) Verb

Parameters

hab (HAB)
(Input). The anchor-block handle returned from a successful call to SPRINIT (Initialize PrintManager). See Appendix E, "System-Specific Information" on page E-1 for more information on SPRINIT (Initialize PrintManager).

PrdName (STRL)
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored if it is **NULL**, a zero-length string, all blanks, or a single asterisk (*), and no validation is done.

For more information on print-descriptor naming conventions, refer to "Using Print Descriptors with the PrintManager Interface" on page 3-3.

Length (ULONG)
(Input). This field is reserved and must have a value of zero.

Buffer (PBYTE)
(Input). This field is reserved.

Usage

Use the SPROPEN (Open) verb to start a print session. You can also use SPROPEN (Open) to define the set of current options by specifying a print descriptor.

For the hab parameter, ensure that you enter the anchor-block handle returned by the SPRINIT (Initialize PrintManager) verb. Otherwise, in the 370 environment PrintManager will abend with an abend code of X'0245' and in OS/400 a value of 0 will be returned.

Return Values

***hprm* (HPRM)**

The SPROPEN (Open) verb should return the session identifier (*hprm*). A return value of 0, however, means an error occurred. Ensure that you save the *hprm* value that the SPROPEN (Open) verb returns. You will need to specify this value on other PrintManager Interface verbs used in a session.

Environment Restrictions

In VM, only one print session can be open at a time. In all other systems, multiple sessions can run concurrently. (See Appendix E, "System-Specific Information" on page E-1 for more information.)

SPRQOPT (Query Option)

MVS	VM	OS/400	OS/2
X	X	X	

Function

List current values for a print option.

Syntax

```
SPRQOPT (hprm, Length, Buffer, success)
```

Figure 4-11. Format of the SPRQOPT (Query Option) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Length (ULONG)
(Input). Specifies the buffer size (see "Usage").

Buffer (BUFFER)
(Input/Output). Returns print-option information. Each language has a specific format for this buffer.

Usage

Use the SPRQOPT (Query Option) verb to list the current values for a print option. Only one print option can be queried for each call to SPRQOPT (Query Option). For more information, refer to Appendix B, "PrintManager C Applications" on page B-1, Appendix C, "PrintManager COBOL Applications" on page C-1, and Appendix D, "PrintManager RPG Applications" on page D-1.

For the hprm parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRSDOC (Start Document)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Starts a print job and specifies a document name for the print job.

Syntax

```
SPRSDOC (hprm, DocName, success)
```

Figure 4-12. Format of the SPRSDOC (Start Document) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

DocName (STRL)
(Input). A string containing the document name for the print job. If no document name is specified, the name is inherited from the previous print job in a session. If no previous document name exists, the operating system provides a default. The *DocName* parameter must follow the system-specific naming conventions. See "Environment Restrictions".

Usage

Use SPRSDOC (Start Document) to start a print job. You must issue SPREDOC (End Document) before issuing another SPRSDOC (Start Document) within the same print session.

If the SPRSDOC (Start Document) verb is successful, PrintManager is placed in Job-in-Progress state.

For the hprm parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means a new print job was initiated, whereas a value of 0 means an error occurred.

Environment Restrictions

DocName is truncated to 10 characters in OS/400. DocName can be an 8 character file name and an 8 character file type (separated by a blank) in VM. DocName is not supported in the MVS environment.

SPRSOPT (Set Option)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Set or change a print option.

Syntax

```
SPRSOPT (hprm, PrdName, Length, Buffer, success)
```

Figure 4-13. Format of the SPRSOPT (Set Option) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

PrdName (STRL)
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored if it is **NULL**, a zero-length string, all blanks, or a single asterisk (*). If *DEL is specified, all print options are deleted from the set of current options, the print descriptor is no longer in effect, and any options specified in **OPTDATA1** are not validated.

For information on naming print-descriptor conventions, refer to *PrintManager Application Programming Interface Reference.*

Length (ULONG)
(Input). Specifies the buffer size.

Buffer (BUFFER)
(Input). The print-option information structure is specified in the OPTDATA1 format. Each language has a specific format for this buffer.

Usage

Use the SPRSOPT (Set Option) verb to validate and set print-option information and to specify a print descriptor (see "Print Option Validation" on page 5-5).

If an error occurs in retrieving a print descriptor, any defaults and validation information from the previous print descriptor are no longer in effect.

If the *PrdName* parameter is specified as **NULL**, a zero-length string, all blanks, or an asterisk, the options specified on this call are validated against the current print descriptor, if one has been previously defined. If no print descriptor has been defined, no validation is performed against a print descriptor.

If a print descriptor is specified after print options have been set, the previously set print options are validated against the new print descriptor.

Note: The last valid print-option value specified overrides previous values specified for that option or the default specified in the print descriptor.

You should specify a print descriptor only once for a print job because each time you specify a print descriptor, it is reapplied and the set of current options is validated.

For the** hprm **parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**

A nonzero value means the print-option values were set successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRWRIT (Write)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Sends a buffer of data to the print job.

Syntax

```
SPRWRIT (hprm, Count, Data, success)
```

Figure 4-14. Format of the SPRWRIT (Write) Verb

Parameters

hprm (HPRM)
(Input). Identifies a valid print session.

Count (LONG)
(Input). The length of the buffer of print data sent to the print job.

Data (BUFFER)
(Input). The buffer of print data sent to the print job.

Usage

Use the SPRWRIT (Write) verb to send a buffer of data to a print job.

If an error occurs, PrintManager returns to Session Active state.

For the hprm parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means the buffer of data was sent to the print job successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

For PrintManager Interface applications, the maximum length of the buffer is 32K bytes in VM.

Chapter 5. Print Options

This chapter:

- Provides an overview of the PrintManager print options
- Discusses print option defaults and validation
- Describes AFP resources.

Overview of PrintManager Print Options

PrintManager defines a set of print options (documented in Appendix A, "PrintManager Print Options" on page A-1) for the SAA environments. The PrintManager Interface supports a subset of these print options in each environment as shown (with an "X") in Table 5-1 and Table 5-2. Table 5-1 lists the print options that you can specify individually, and Table 5-2 lists those options whose values are used to build an inline form definition (when the FORMDEF option has a value of *BUILD).

The subset of options that the PrintManager Interface supports is determined by the printing capabilities of each SAA environment. All PrintManager options, however, are supported by the API because they can be set in a print descriptor and queried to determine the capabilities of a print system component.

Appendix A, PrintManager Print Options also documents a set of values for the defined options, but you can modify these values or define additional values for your organization's needs. If you want, you can also define additional print options. Any additional values and options you define, however, are ignored by the PrintManager Interface. Installation defined options or values may be used to specify printing for or provide information about print system components that are unique to that installation.

You can specify print options in an application written to the PrintManager Interface or in a print descriptor. The IBM SAA PrintManager PRF also provides a set of options that are consistent with those provided with the PrintManager Interface. For more information, refer to the *IBM SAA PrintManager User's Guide*.

Table 5-1 (Page 1 of 2). CPI Print Options by Environment				
Option	**MVS**	**VM**	**OS/400**	**OS/2**
BACKGRNDMIX				
BARCODESET				
CAPSGRAPHIC				
CAPSRASTER				
CC	X	X	X	
CKPTLINE	X			
CKPTPAGE	X	X		
CKPTSEC	X			
CLASS	X	X		
COLORNUM				
COPIES	X	X	X	
DATACK	X	X	X	
DATATYPE	X	X	X	
DEVDRIVERTYPE				
DEVMODEL				
DNLDFONT				
DOCOWNER		X		
DUPLEX		X	X	
FCB	X	X		
FIDELITY			X	
FLASHCNT	X	X		
FLASHNAME	X	X		
FOCASET				
FONT	X	X	X	
FOREGRNDMIX				
FORM	X	X	X	
FORMDEF	X	X	X	
GDDMDEVTOKEN	X	X		
GRAPHICSET				
HORIZRES				
IMAGESET				
INBIN		X	X	
INDEX	X			
LINDEX	X			
LINECT	X			
LPI				
MEDIATTRIBUTES				
MEDIATYPE				

Table 5-1 (Page 2 of 2). CPI Print Options by Environment				
Option	**MVS**	**VM**	**OS/400**	**OS/2**
MEDIAXLEFTCLIP				
MEDIAXPELS				
MEDIAXRIGHTCLIP				
MEDIAXWIDTH				
MEDIAYBOTTOMCLIP				
MEDIAYHEIGHT				
MEDIAYPELS				
MEDIAYTOPCLIP				
MESSAGES	X	X		
MODCASET	X	X		
MODIFYNAME	X	X		
MODIFYTRC	X	X		
OUTDISP	X	X	X	
OUTMETHOD	X			
OUTPUTID	X	X	X	
OUTQ			X	
OVERLAY	X	X	X	
PAGEDEF	X	X	X	
PAGERANGE			X	
PAGESEG	X	X	X	
PRMODE	X	X	X	
PRTENVIRONMENT				
PRTY	X		X	
RSCSID		X		
SCHEDULE			X	
TECHNOLOGY				
TEXTSET				
TRC	X	X	X	
UCS	X			
VERTRES				
WRITER	X			

Table 5-2. CPI Print Options Used to Build Form Definitions				
Option	**MVS**	**VM**	**OS/400**	**OS/2**
DUPLEX	X	X	X	
FLASHNAME	X	X	X	
INBIN	X	X	X	
JOGOUT	X	X	X	
LEFTMAR	X	X	X	
OVERLAY	X	X	X	
PRTDIRECTION	X	X	X	
PRTQUAL	X	X	X	
TOPMAR	X	X	X	

Print Option Defaults and Validation

By using print descriptors, you can establish default values and valid values for print options. Default values provide job and printer defaults. Valid values in a print descriptor are used to validate print options and values. Print descriptors can be specified using the SPROPEN (Open) verb and SPRSOPT (Set Option) verb. This allows print options to be validated before a print job is submitted to the spool. The following sections discuss print option defaults and validation. For more information on creating print descriptors, see the *PrintManager Application Programming Interface Reference*.

Print Option Defaults

When you specify a print descriptor, you inherit the print-option defaults specified in that print descriptor. If the print descriptor contains references, these references can be used to merge print-option information from the referenced print descriptors. Refer to the *PrintManager Application Programming Interface Reference* for information on merging print options.

For example, if a print descriptor specifies a default of 10 copies for a printer, any application that specifies the print descriptor to the PrintManager Interface when submitting a print job will have 10 copies as a default value. Similarly, if a print descriptor is used to specify one copy as a default for a class of jobs (such as a standard memo), any application that specifies this print descriptor will have one copy as the default.

An application can override the default values specified in a print descriptor using SPRSOPT (Set Option), but only within the valid values specified in the validation rules in the print descriptor. For example, if a print descriptor specifies a default value of one copy, and a range of valid values from one to ten copies, an application could specify five copies to override the default, but not fifteen copies. For more information, see "Print Option Validation" on page 5-5.

The set of current options contains a combination of the default values inherited from the print descriptor and the print-option values overridden by calls to SPRSOPT (Set Option).

Note: See "Using Print Descriptors with the PrintManager Interface" on page 3-3 for information on the advantages of using print descriptors.

Print Option Validation

You can explicitly set print options using SPRSOPT (Set Option). Print options that are explicitly set are validated using the current print descriptor (if one is specified).

If a print descriptor is specified, PrintManager validates print options as follows:

1. If the option exists in the print descriptor, it is validated using the rules and valid values in that print descriptor. If the value is valid, it is set and PrintManager updates the set of current options with the new value.
2. If the print option does not exist in the print descriptor, the print option is invalid and PrintManager does not update the set of current options.
3. If a current print descriptor exists and you specify a new print descriptor on SPRSOPT (Set Option), PrintManager validates the current options and values using the new print descriptor as follows:
   a. The new print descriptor establishes a new set of print options and default values.
   b. All print options that were explicitly set are validated using the rules of the new print descriptor. PrintManager validates these print options and values as follows:
      - If the current option is not in the new print descriptor, the print option is deleted from the set of current options.
      - If the current value is not valid, the new default value is used (if one exists); otherwise, the print option has no value.
4. PrintManager returns errors for invalid print options and print-option values specified in an application. However, if a valid value is specified after an invalid value for the same option, the valid value is added to the set of current options. If an invalid value is specified after a valid value for the same option, the previous valid value remains in the set of current options.

Whether or not a print descriptor is specified, some print options are also validated according to operating system limitations. For example in OS/400, if you specify a value of *MYTYPE* for DATATYPE, this will result in an error because OS/400 does not support this data type.

Appendix A, PrintManager Print Options lists the print options and validation rules defined by PrintManager.

Validation rules are specified in a print descriptor and are used by PrintManager as follows:

Rule	Description
NOVALIDATION	Validation does not occur and any value is valid.
STRING	Valid values are a string within the maximum length defined by the print descriptor. Values cannot be zero length, and may be any alphanumeric character except the null character.
RANGE	Valid values consist of a numeric value that falls within the range that is defined in the print descriptor. The valid range is defined in the print descriptor as a minimum and maximum value, as well as the number of significant decimal places.

LIST The list of valid values is defined in the print descriptor. For a specific value to be valid, it must appear on this list. Validation of list values is *not* case sensitive.

The format of numeric values supported by PrintManager is as follows:

- The value of the decimal character is a period(.).
- The negative indicator is a minus sign (-). There is no trailing negative indicator.
- There is no thousands separator character.

Any string or part of a string enclosed within a pair of double quotes (" ") will be treated as a literal part of a value. This allows a comma, a blank, a semicolon, an equal sign, or a double quote (when enclosed by paired double quotes) to be used as part of a value. If no matching double quote is found, the end of the string is assumed to be the zero terminator for C language, and the user specified length in any other language.

For a validation rule of LIST, the values are validated using a character-for-character compare. For example, if the list of valid values is YES, NO, MAYBE, the value of YES will be valid, but the value of "YES" will not be valid. If "YES" is supposed to be valid, it must be included in the list of valid values (YES, "YES", NO, MAYBE).

For a validation rule of LIST, validation is not case sensitive. For example, if the valid value specified in the print descriptor is YES, application specified values of YES, yes, Yes, and yeS are among the possible valid values. Similarly, the same values would be valid if the print descriptor specified a valid value of yes. As a result, the print descriptor need only specify a single valid value (in any combination of case) to ensure that all combinations of case specified in an application are valid. Conversely, you cannot restrict a value to a particular combination of case. For example, valid values of YES, yes, Yes in a print descriptor will not exclude yES as a valid value.

For the validation rules NOVALIDATION and STRING, double-byte character strings can be specified as part of a value. Values can contain single-byte character strings, double-byte characters strings, or both. However, double-byte character strings must be an even number of bytes and EBCDIC double-byte character strings must be delimited by the shift-out and shift-in characters. For validation rules of STRING, the maximum length defines the number of bytes.

AFP Resources

On some of the options described in Appendix A, PrintManager Print Options, you can specify the AFP resources you need for a print job. AFP resources include form definitions, page definitions, fonts, overlays (electronic forms), and page segments (graphic images). AFP resources and the corresponding print options are as follows:

- **Form definitions**. Form definitions contain information about the printing medium (for example, a sheet of standard memo paper) and how it is handled. You can use a form definition to specify number of copies, overlays for the job, duplexing or simplexing, and the starting point for placement of data on the page.

  You specify the form definition on the **FORMDEF** print option. You can also build a simple form definition to be created when the job is written to the spool (refer to Table 5-2 on page 5-4 for a list of the print options whose values you can use to build a form definition).

  Note: If you build a form definition but do not specify print options, a default form definition with a page offset of (0, 0) and no other specifications will be built.

- **Page Definitions**. Page definitions contain information about the placement and formatting of line data on the print medium. For example, you can use page definitions to specify line length, page length, fonts for a print job, and rotated printing. You specify a page definition on the **PAGEDEF** print option.

- **Overlays** (electronic forms). Overlays are collections of coded information that describe where to put boxes, lines, shading, text, logos, and graphic images on forms. When printed with variable application data, overlays can replace the need for preprinted forms. You specify overlays on the **OVERLAY** print option.

- **Fonts**. Fonts are a family or assortment of characters in a given size and style. You specify fonts on the **FONT** print option.

- **Page segments**. Page segments can contain text, raster graphic data, or both. Examples of page segments include logos, signatures, bar charts, and drawings. Page segments can be printed anywhere on a page or at the same place in every page of a print job. The print data must specify page segments and their placement. You specify page segments on the **PAGESEG** print option.

When specifying AFP resources in print options:

- You can use PrintManager print options to specify AFP resources only if **DATATYPE** = **MODCAP**, **AFPDS**, **AFPDSLINE**, or **LINE**.

- If you specify multiple values on the **FONT**, **OVERLAY**, or **PAGESEG** options, each value must be preceded by the appropriate keyword. For example, in VM you might specify:

  FONT *CHAR GT10 ***CHAR** GT12
  OVERLAY *INLINE O1S16 OVLY3820 A ***SYSTEM** O11040

- You can specify only one value on the **FORMDEF** and **PAGEDEF** options.

- In addition, you must be aware of how your printer driver processes the AFP resources that you specify with PrintManager. For example, if you are using PSF/VM as your printer driver and you specify an inline overlay in your application, that overlay will not be used unless you also specify a form

definition which names that overlay. For example, you may want to specify that a form definition is built as follows:

OVERLAY *INLINE O1596 OVLY3820 A ***FORMDEF** O1596
FORMDEF *BUILD

Refer to your printer-driver documentation for more information on how AFP resources are processed.

Inline Resources

Depending on the system, you can usually specify that AFP resources will be placed *inline* with a print job as the job is written to the spool. You do this by using the ***INLINE** keyword and specifying a fully qualified resource name on the print option. Inline resources can be routed to another system because they are "packaged" with the print job and do not have to be sent separately or reside on the remote system.

For example, suppose that a TSO user wants to print a job on a VM system using a particular overlay. A PrintManager application or print descriptor can specify that the overlay will be placed inline with the print job as the job is written to the TSO spool; therefore, the overlay does not have to reside on the VM system.

When specifying inline AFP resources, however, you must be aware of whether inline resources are supported on the system that will print the job. Refer to your printer-driver documentation for information on the support for inline AFP resources.

Appendix A. PrintManager Print Options

The options in this appendix are consistently defined across the environments supported by PrintManager, although they may not be supported by PrintManager in all environments (see "Environment Notes"). All options, however, can be set and queried in all environments. This appendix describes the option values provided with PrintManager, but you can modify these values or define additional values for your organization's needs. If you want, you can also define additional print options. Any additional values and options you define, however, are ignored by the PrintManager Interface.

Print-option information is described in this format:

- Option name and description
- Validation rules and valid values
- Environment notes (environment support and use).

Using PrintManager Print Options for AFP

If you are using AFP, you must understand how the printer driver will use the PrintManager print option values that you specify. For more information, see the publications listed under "For Printing" on page 1-8. For a summary of how the PrintManager print options correspond to operating system commands and parameters, see Table A-1 on page A-23.

To successfully distribute print jobs between MVS and VM systems, the print options you specify must be supported by PrintManager in both environments. See "Environment Notes" for information on the print options you want to specify and Table 5-1 on page 5-2 and Table 5-2 on page 5-4 for a summary of the options that are valid in both environments. Although PrintManager provides a full range of print options, typically you need only a small set of these options for a specific type of print job.

OPTION

BACKGRNDMIX Specifies the background-mix toning supported by the printer.

Validation Rule: **LIST**

Valid Values:

OR	Logical OR
OVERPAINT	Overpaint
XOR	Logical XOR
LEAVEALONE	Leave alone

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

BARCODESET Specifies the MO:DCA-P bar code capability supported by the printer driver.

Validation Rule: **LIST**

Valid Values:

NONE	No bar code support

BCD1 BCD1 support

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

CAPSGRAPHIC Specifies the graphics capability supported by the printer.

Validation Rule: **LIST**

Valid Values:

KERNING Kerning is supported.

OUTLINEDEF Printer has default outline font.

IMAGEDEF Printer has default image font.

MARKERSDEF Default markers will be scaled.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

CAPSRASTER Specifies the raster capabilities supported by the printer.

Validation Rule: **LIST**

Valid values:

BITBLT GpiBitBlt is supported.

BANDING Printer supports banding.

SCALING Printer supports scaling.

SETPEL GpiSetPel supported.

FONTS Printer supports raster fonts.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

CC Specifies the type of carriage control characters contained in a data stream.

Validation Rule: **LIST**

Valid Values:

NO Does not contain carriage control characters.

YES Contains ANSI carriage control characters.

MACHINE Contains machine carriage control characters.

Environment Notes: This option is functionally supported by a printer driver in the MVS, VM, and OS/400 operating systems. If **DATATYPE** is **MODCAP**, **AFPDS**, or **AFPDSLINE** and **CC** is not set or set to NO, **CC** is changed to YES for VM and OS/400 or MACHINE for MVS.

CKPTLINE Specifies the maximum number of lines in a logical page. JES also uses **CKPTLINE** with **CKPTPAGE** to determine when to take data-set checkpoints. Checkpoints are used to save all information necessary to resume printing if an error occurs.

Note: If you do not set a value for this option, a JES or PSF default will be used.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 32767 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system.

CKPTPAGE

Specifies the number of logical pages to be printed before taking a checkpoint. Checkpoints are used to save all information necessary to resume printing if an error occurs.

Note: If you do not set a value for this option, a JES or PSF default will be used.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 32767 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the MVS and VM operating systems. On MVS, if you do not specify a value, an installation-defined default is used.

CKPTSEC

Specifies the number of seconds to elapse before taking a checkpoint. Checkpoints are used to save all information necessary to resume printing if an error occurs.

Note: If you do not set a value for this option, a JES or PSF default will be used.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 32767 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system. If you specify both **CKPTPAGE** and **CKPTSEC**, the **CKPTSEC** value is used unless the installation has specified that checkpointing is based only on pages.

CLASS

Specifies the print output class (a printer or group of printers) for job scheduling.

Validation Rule: **LIST**

Valid Values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, *

Note: These valid values make up the entire set of values PrintManager supports for **CLASS**, but your organization may define a subset of these values for actual use.

Environment Notes: This option is functionally supported by a printer driver in the MVS and VM operating systems.

COLORNUM

Specifies the number of distinct colors supported by the printer, including reset (grey scales count as distinct colors).

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

COPIES

Specifies how many copies of a print job will be printed.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 255 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

DATACK

Specifies which printer errors are reported. Two types of data check errors can occur. Print-positioning errors result from attempts to print outside the valid printable area. Invalid-character errors result from attempts to use a code point that is not assigned to a character.

If print-positioning errors are reported (UNBLOCK or BLKCHAR values), your printer may provide exception highlighting to mark the location of the error. For more information, refer to your printer documentation or *Advanced Function Printing: Printer Information*.

Note: If you do not set a value for this option, a PSF or OS/400 system default will be used.

Validation Rule: **LIST**

Valid Values:

BLOCK	Block (do not report) print-positioning errors and invalid-character errors. Printing continues but data may be lost.
UNBLOCK	Report all errors.
BLKCHAR	Block only invalid-character errors. Print-positioning errors are reported normally.
BLKPOS	Block only print-positioning errors. Invalid-character errors are reported normally.

Environment Notes: This option is functionally supported by a printer driver in the MVS, VM, and OS/400 operating systems. In OS/400, the value for this option:

- Is set to UNBLOCK if the **FIDELITY** option is set to ABSOLUTE.
- Is changed to BLOCK at print time if the **FIDELITY** option is set to CONTENT and the print file is destined for a twinax attached printer.

DATATYPE

Identifies the data stream type of the print job.

Validation Rule: **LIST**

Valid Values:

AFPDS	An AFPDS data stream (with X'5A' carriage control characters)
AFPDSLINE	A mixed data stream (AFPDS and line data)
ASCII	An IBM ASCII data stream
DBCS_ASCII	An IBM double-byte character set ASCII data stream
LINE	Either a line data stream (for example, 1403) or unformatted data. If the data contains carriage control characters, **CC** must also be set to YES or MACHINE.

MODCAP	Any MO:DCA data stream
PCL	An HP data stream
PM_Q_RAW	A passthrough data stream
PM_Q_STD	The standard Presentation Manager data stream
POSTSCR	A PostScript data stream
SCSSNA	An SNA SCS data stream (370 environment)
SCS400	An OS/400 SCS data stream

Environment Notes: Supported in the VM, MVS, and OS/400 environments as follows:

- In VM, MVS, and OS/400, if **DATATYPE** is **MODCAP**, **AFPDS**, or **AFPDSLINE** and **CC** is not set or set to NO, **CC** is changed to YES for VM and OS/400 or MACHINE for MVS.
- In VM and MVS you can place any of the above data streams on the spool.
- In OS/400, you can place the **AFPDSLINE**, **LINE**, **MODCAP**, **AFPDS**, **ASCII**, and **SCS400** data streams on the spool. PrintManager assumes a default of **SCS400** in OS/400.

  For the **AFPDSLINE** and **LINE** data streams, a PrintManager Interface application must either:

  - If using the SPRWRIT (Write) verb, write the data a line at a time
  - If using the SPRADDF (Add File) verb, specify a file that contains either a single line or a single AFPDS structured field per record.

  In OS/400, If the data is line data containing double byte characters, the **PRMODE** option must be set to SOSI1 or SOSI2 and your system must be an IGC machine.

DEVDRIVERTYPE Identifies the printer driver.

Validation Rule: **LIST**

Valid Values:

PSF	Print Services Facility driving IPDS printers, for example, the IBM 3800 Printing Subsystem, IBM 3820 Page Printer, or IBM 3827 Page Printer.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

DEVMODEL Identifies the type and model number of the printer.

Validation Rule: **NOVALIDATION**

Valid Values: Any character string that represents the type and model number of the printer.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

DNLDFONT Specifies the number of fonts that can be downloaded to the printer at any one time.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

DOCOWNER Specifies the print-job owner.

Validation Rule: **NOVALIDATION**

Valid Values: Any character string representing the print-job owner in the following format:

FORMAT

userid The userid of the user.

Environment Notes: This option is functionally supported in the VM operating system. This value correlates to the FOR parameter on the CP SPOOL command. This value indicates the userid under which printed or punched output is produced.

In VM, **RSCSID** and **DOCOWNER** are mutually exclusive. If both are specified, the **DOCOWNER** value will be used. If the print job must be directed to a remote system using **RSCSID**, **DOCOWNER** must not be specified.

DUPLEX Specifies how the job will be printed on the media (for example, simplexing or duplexing). Values from this option are used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

Validation Rule: **LIST**

Valid Values:

NO Simplexed output.

YES Two-sided output.

TUMBLE Tumble two-sided output.

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems. In VM and OS/400, you can specify **DUPLEX** without specifying ***BUILD** for the **FORMDEF** option. In OS/400, the value specified for **DUPLEX** will override any value specified in the active form definition.

FCB Specifies the name of a forms control buffer (FCB) for formatting line data for a 3800-1 printer.

Validation Rule: **STRING**

Valid Values: 4 (maximum length)

Environment Notes: This option is functionally supported by a printer driver in the MVS and VM operating systems.

FIDELITY Specifies if the printer driver will try to print a file that contains errors.

Validation Rule: **LIST**

Valid Values:

ABSOLUTE The job is printed only if the file can be printed exactly as specified by the data stream and external controls. If **DATACK** is set, the **DATACK** value is ignored and the system sets it to UNBLOCK.

CONTENT The file is printed if at all possible. All available exception handling is used.

Environment Notes: This option is functionally supported in OS/400 (only AFP).

FLASHCNT

Specifies the number of pages to print with the forms flash specified in the **FLASHNAME** option.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 255 (maximum)

Environment Notes: This option is functionally supported by the MVS and VM operating systems. AFP printer drivers may provide different function support than the operating systems. For more information, refer to your printer driver documentation.

FLASHNAME

Specifies the name of a forms flash for printing photographic images on an IBM 3800. For information on forms flashing, refer to *Forms Design Reference for the 3800*.

Validation Rule: **STRING**

Valid Values: 4 (maximum length)

Environment Notes: This option is functionally supported by a printer driver in the MVS and VM operating systems.

FOCASET

Specifies the SAA font capability supported by the printer driver.

Validation Rule: **LIST**

Valid Values:

NONE No FOCA font support.

FOCA2 FOCA2 fonts supported.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

FONT

Specifies one of the following:

- The location of fonts that are included in a line, page, or mixed data stream
- The names of fonts to be associated with table reference characters (TRCs) in a document.

Validation Rule: **NOVALIDATION**

Valid Values: Any string specifying the fonts for the job in one of the following formats:

***INLINE name** This keyword is supported in MVS and VM. For only AFP print jobs, use ***INLINE** to select a font from a system library (on the system where the print job is submitted) and place it inline with the print job. Refer to your printer-driver documentation for

information on how inline resources are handled in each environment. The *name* field is the fully qualified name of the coded font.

In VM, if the *name* field is not a fully qualified name, the default file type is based on the two-character prefix of the file name. If the prefix is "X0," the default file type is FONT3820. If the prefix is "X1," "X2," "X3," or "X4," the default file type is FONT38PP. In VM, a file mode of "A" is assumed. To search all accessed disks, "*" can be specified as a wild card.

***SYSTEM name**

This keyword is supported in MVS and VM. For only AFP print jobs, use ***SYSTEM** to select a font (specified in *name*) from a system library on the system where the print job is to be printed. The *name* field is the 1–8 character name of a font. If the font name is not specified, a system default is used.

For VM only, the *name* field can also include an optional 1–8 character file type. This file type is based on the two-character prefix of the *name* field. If the prefix is "X0," the default file type is FONT3820. If the prefix is "X1," "X2," "X3," or "X4," the default file type is FONT38PP.

***CHAR name**

This keyword is supported in MVS, VM, and OS/400. For line data print jobs with TRC controls, use the ***CHAR** *name* field to specify up to four font names or (for 3800 line printers) character-arrangement tables that are used for the print job. If the print file does not contain table reference characters (TRCs), only the first font is used. If the print file contains TRCs, the **TRC** option must be set to YES and the fonts must be specified in the order that corresponds to the TRC values. The *name* field must be 1–4 characters.

In OS/400, the object type is "*FNTRSC."

You can specify multiple values for the **FONT** option by entering multiple keywords and values separated by blanks. For example: ***CHAR** GT10 ***CHAR** GB10

In VM and MVS, to specify the name of a font for a line data print file and to specify that the font resides in a system library, use: ***CHAR** *name* ***SYSTEM** *name*, where the name field is the same for both keywords. ***INLINE** can also be paired with ***CHAR** to specify that the font to be used will be included inline with the print file.

Environment Notes: Refer to the environment information above.

FOREGRNDMIX Specifies the foreground-mix toning supported by the printer.

Validation Rule: **LIST**

Valid Values:

OR	Logical OR
OVERPAINT	Overpaint
XOR	Logical XOR
LEAVEALONE	Leave alone
AND	Logical AND
SUBTRACT	(inverse source) AND dest
MASKRCNOT	Source AND (inverse dest)
ZERO	All zeros
NOTMERGSRC	Inverse (source OR dest)
NOTXORSRC	Inverse (source XOR dest)
INVERT	Inverse (dest)
MERGESRCNOT	Source OR (inverse dest)
NOTCOPYSRC	Inverse (source)
MERGENOTSRC	Inverse (source) OR dest
NOTMASKSRC	Inverse (source AND dest)
ONE	All ones

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

FORM Specifies an installation-defined forms code that specifies the media for a print job.

Validation Rule: **NOVALIDATION**

Valid Values: Any character string specifying the forms code.

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

FORMDEF Specifies the form definition for an AFP print job.

Validation Rule: **NOVALIDATION**

Valid Values: Any string specifying the form definition in the following formats (all keywords are supported in VM, MVS, and OS/400):

***INLINE name**

Use ***INLINE** to select a form definition from a system library (on the system where the print job is submitted) and place it inline with the print job. Refer to your printer-driver documentation for information on how inline resources are handled in each environment. The *name* field is the fully qualified name of the file containing the form definition.

In VM, if the *name* field is not a fully qualified name, the default file type is FDEF38PP and a file mode of "A" is assumed. To search all accessed disks, "*" can be specified as a wild card.

In OS/400, the object type is "*FORMDF."

***SYSTEM name**

Use ***SYSTEM** to select a form definition (specified in *name*) from a system library on the system where the print job is to be printed. The *name* field is the 1–8 character name of a form definition. ***SYSTEM** is also used to name a form definition that is already a part of the print data.

For VM only, the *name* field can also include an optional 1–8 character file type. In VM, the default file type is FDEF38PP.

In OS/400, the object type is "*FORMDF."

***BUILD**

Use ***BUILD** to specify that a form definition will be built and placed inline using values from the following print options: **TOPMAR**, **LEFTMAR**, **INBIN**, **PRTDIRECTION**, **DUPLEX**, **PRTQUAL**, **OVERLAY**, **JOGOUT**, **FLASHNAME**.

Environment Notes: Refer to the environment information above.

GDDMDEVTOKEN

Specifies a GDDM device token. It is required by PrintManager for printing documents where the data stream must be transformed from MO:DCA-P format to AFPDS format. This option is valid only when **DATATYPE** = **MODCAP** and **MODCASET** = **AFPDS**. For more information on the MO:DCA-P to AFPDS data stream transform, see to "Printing MO:DCA-P Documents in VM and MVS" on page E-7.

Validation Rule: **LIST**

Valid Values: Any list of the valid GDDM device tokens for GDDM family 4 devices. For more information, refer to your GDDM documentation.

Environment Notes: This option is functionally supported by a printer driver in the MVS and VM operating systems.

GRAPHICSET Specifies the MO:DCA-P graphics capability supported by the printer driver.

Validation Rule: **LIST**

Valid Values:

NONE	No GOCA graphics support.
DR2V0	Identifies GOCA DR2V0 graphics support.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

HORIZRES Identifies the horizontal resolution of the specified device in pels per meter.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

IMAGESET Specifies the MO:DCA-P image capability supported by the printer driver.

SYSTEM **VALID VALUES**

Validation Rule: **LIST**

Valid Values:

NONE	No image support
IMD1	IM1 images supported
FS10	IOCA FS10 images supported.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

INBIN Specifies the source for paper or other media for a print job. Values from this option are used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 255 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems. In VM and OS/400, you can specify **INBIN** without specifying ***BUILD** for the **FORMDEF** option. In OS/400, the **INBIN** value overrides any value specified in the active form definition.

INDEX For 3211 printers with the indexing feature only, specifies the number of print positions for the left margin indentation.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 31 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system.

JOGOUT Specifies jogging (offsetting) of copy groups within a print job in the printer output bin. For continuous forms printers that do not burst output into cut sheets, specifies that edge marking is to change on copy group boundaries to facilitate output separation. Values from this option are only used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

Validation Rule: **LIST**

Valid Values:

YES	Jog output.
NO	Do not jog output.

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

LEFTMAR

Specifies the width of the left margin in pels. Values from this option are only used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 32767 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

LINDEX

For 3211 printers with the indexing feature only, specifies the number of print positions for the right margin indentation.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 31 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system.

LINECT

For a line printer, specifies the maximum number of lines JES2 is to print on each output page.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 255 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system.

LPI

Specifies the number of printed lines per inch.

Validation Rule: **RANGE**

Valid values: 0 (decimal places), 1 (minimum), 99 (maximum)

Environment Notes: This option is used by the PRF component of IBM SAA PrintManager to select a page definition for line printing.

MEDIATTRIBUTES

Specifies media availability.

Validation Rule: **LIST**

Valid Values:

HCAPS_CURRENT	Current form or media is installed.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIATYPE

Specifies the type of media (for example, continuous-forms paper or cut-sheet paper) being used by the printer.

Validation Rule: **LIST**

Valid Values:

CONTINUOUS	Continuous-forms paper.

SHEET Cut-sheet paper.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAXLEFTCLIP Specifies the left clip limit of the media in millimeters.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAXPELS Specifies the number of pels between the left and right clip limits of the media.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAXRIGHTCLIP Specifies the right clip limit of the media in millimeters.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAXWIDTH Specifies the width (left to right) of the media in millimeters.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAYBOTTOMCLIP

Specifies the bottom clip limit of the media in millimeters.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAYHEIGHT Specifies the height (top to bottom) of the media in millimeters.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAYPELS Specifies the number of pels between the top and bottom clip limits of the media.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MEDIAYTOPCLIP Specifies the top clip limit of the media in millimeters.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

MESSAGES

For a job sent to an AFP printer, specifies the number of errors to be received before terminating the job (the number of errors received may not equal the number of messages received). Setting a high value for this option helps to debug a document, whereas setting a low value ensures that print job processing will stop if an error is detected.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 1000 (maximum). A value of 0 specifies no messages will be received. A value of 1000 specifies that all messages will be received.

Environment Notes: This option is functionally supported by a printer driver in the MVS and VM operating systems.

MODCASET

Identifies the level of MO:DCA-P supported by the printer driver. This print option is used to determine if the data stream format is to be converted before placing the print job on the spool. For more information on the MO:DCA-P to AFPDS data stream transform, see to "Printing MO:DCA-P Documents in VM and MVS" on page E-7.

Validation Rule: **LIST**

Valid Values:

FS9	Specifies the Interchange Set 1 subset of the MO:DCA-P architecture that is defined as SAA print interchange format.
AFPDS	Identifies the subset of the MO:DCA-P architecture that is supported by PSF/MVS and PSF/VM.
AFPDSE1	Identifies the subset of the MO:DCA-P architecture that includes Interchange Set 1 and other extensions, such as, IM Image, bar codes, and page definitions.

Environment Notes: This option is meaningful only for AFPDS data streams, and is used by PrintManager to determine if data stream transforms are required in the MVS and VM environments.

MODIFYNAME

Specifies the copy-modification module to be used to print a job on a 3800 printer.

Validation Rule: **STRING**

Valid Values: 4 (maximum length)

Environment Notes: This option is functionally supported by the MVS and VM operating systems.

MODIFYTRC

Specifies the character-arrangement table to be used with a copy-modification module (specified in **MODIFYNAME**) on a 3800 printer.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 3 (maximum)

Environment Notes: This option is functionally supported by the MVS and VM operating systems. In VM and MVS, if **MODIFYTRC** is specified, **MODIFYNAME** must be specified and must have a valid value.

OUTDISP Specifies the disposition of the print job.

Validation Rule: **LIST**

Valid Values:

HOLD	Specifies that the print job will be held until released.
NOHOLD	Specifies that the print job will not be held.
KEEP	Specifies that the print job will be saved after it is printed.
NOKEEP	Specifies that the print job will not be saved after it is printed.

Note: You can specify both HOLD and KEEP, and you can specify NOHOLD and NOKEEP.

Environment Notes: HOLD and NOHOLD are supported in the MVS, VM, and OS/400 environments. KEEP and NOKEEP are supported only in OS/400.

OUTMETHOD Specifies postprocessing operations done by the printer.

Validation Rule: **LIST**

Valid Values:

BURST	Burst continuous-forms output.

Environment Notes: This option is functionally supported only in the MVS environment.

OUTPUTID Specifies the printer name.

Validation Rule: **NOVALIDATION**

Valid values: Any character string specifying the printer name in one of the following formats:

FORMAT

nodeid.outputid	The network nodeid and the destination (for VM) or the userid (for MVS) of the printer. Use this format when the user is on a network node different from the node where the job is to be processed. In VM, if this format is used, the **RSCSID** print option should be used to specify the userid of the local RSCS service machine. Otherwise, the user's virtual printer device (00E) must be spooled to the userid of the local RSCS service machine before running the print application, or the user must transfer the print job to the userid of

the local RSCS service machine after running the print application.

outputid The userid of the printer. Use this format when the user is on the same network node as the node where the job will be processed. In OS/400, this value is also used as the output queue name if it is not specified in the **OUTQ** option.

Environment Notes: This option is functionally supported by the spooling program in the MVS, VM, and OS/400 operating systems, and it correlates to the operating system values described in Table A-1 on page A-23.

OUTQ

Specifies the output queue for the spooled output file.

Validation Rule: **STRING**

Valid Values: 10 (maximum length)

Environment Notes: This option is functionally supported by a printer driver in the OS/400 operating system.

OVERLAY

Specifies the electronic overlays for a print job.

Validation Rule: **NOVALIDATION**

Valid Values: Any character string specifying up to 8 overlays for the print job in one of the following formats:

***INLINE name** This keyword is supported in VM, MVS, and OS/400. Use ***INLINE** to select an overlay from a system library (on the system where the print job is submitted) and place it inline with the print job. Refer to your printer-driver documentation for information on how inline resources are handled in each environment. In addition, the overlay must be named in the data stream in order to print. The *name* field is the fully qualified name of the file containing the overlay.

In VM, if the *name* field is not a fully qualified name, the default file type is OVLY38PP and a file mode of "A" is assumed. To search all accessed disks, "*" can be specified as a wild card.

In OS/400, the object type is "*OVL."

***SYSTEM name** This keyword is supported in MVS and VM. Use ***SYSTEM** to select an overlay (specified in *name*) from a system library on the system where the print job is to be printed. In addition, the overlay must be named in the data stream in order to print. The *name*

field is the 1 – 8 character name of an overlay. If the overlay name is not specified, a system default is used.

For VM only, the *name* field can also include an optional 1 – 8 character file type. The default file type is OVLY38PP.

***FORMDEF name** This keyword is supported in VM, MVS, and OS/400. Use **FORMDEF** to specify that the overlay in the *name* field will be specified in a form definition built with the ***BUILD** keyword. The location of the overlay used is determined by the printer driver. This overlay will be included on every page of the document. The *name* field is the 1 – 8 character name of the overlay. Values from this option are only used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

You can specify multiple overlays by entering multiple keywords and names separated by blanks. For example: **OVERLAY *SYSTEM** 01IBM ***SYSTEM** 01SYS

To specify that an overlay be named in an inline form definition (built with the ***BUILD** keyword of the **FORMDEF** option) and that the overlay resides in a system library, use: **OVERLAY *FORMDEF** *name* ***SYSTEM** *name*, where the name field is the same in both keywords.

***INLINE** can also be paired with ***FORMDEF** to specify that for the overlay is to be named in an inline form definition and that the overlay will be included inline with the print file.

Environment Notes: Refer to the environment information above.

PAGEDEF Specifies the name and location of the page definition used for AFP printing of a line data print job.

Validation Rule: **NOVALIDATION**

Valid Values: Any character string that specifies the page definitions for the print job in one of the following formats (both keywords are supported in VM, MVS, and OS/400):

***INLINE name** Use ***INLINE** to select a page definition from a system library (on the system where the print job is submitted) and place it inline with the print job. Refer to your printer-driver documentation for information on how inline resources are handled in each environment. The *name* field is the fully qualified name of the file containing the page definition.

In VM, if the *name* field is not a fully qualified name, the default file type is

PDEF38PP and a file mode of "A" is assumed. To search all accessed disks, "*" can be specified as a wild card.

In OS/400, the object type is "*PAGDFN."

***SYSTEM name** Use ***SYSTEM** to select a page definition (specified in *name*) from a system library on the system where the print job is to be printed. The *name* field is the 1–8 character name of a page definition. ***SYSTEM** is also used to name a page definition that is already a part of the print data.

For VM only, the *name* field can also include an optional 1–8 character file type. The default file type is PDEF38PP.

In OS/400, the object type is "*PAGDFN."

Environment Notes: This option is functionally supported by a printer driver in the MVS, VM, and OS/400 operating systems. Refer to the environment information above.

PAGERANGE Specifies the beginning page and ending page of a print job.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 2^{31} (maximum)

Environment Notes: This option is functionally supported only in the OS/400 environment.

PAGESEG Specifies the location of the page segments for a print job. The page segments must be named in the data stream in order to print.

Validation Rule: **NOVALIDATION**

Valid Values: Any character string specifying the page segments for the job in one of the following formats:

***INLINE name** This keyword is supported in VM, MVS, and OS/400. Use ***INLINE** to select a page segment from a system library (on the system where the print job is submitted) and place it print job. Refer to your printer-driver documentation for information on how inline resources are handled in each environment. The *name* field is the fully qualified name of the file containing the page segment.

In VM, if the *name* field is not a fully qualified name, the default file type is PSEG38PP and a file mode of "A" is assumed. To search all accessed disks, "*" can be specified as a wild card.

In OS/400, the object type is "*PAGSEG."

***SYSTEM name** This keyword is supported in MVS and VM. Use ***SYSTEM** to select a page segment (specified in *name*) from a system library on the system where the print job is to be printed. The *name* field is the 1–8 character name of a page segment. If the page segment name is not specified, a system default is used.

For VM only, the *name* field can also include an optional 1–8 character file type. The default file type is PSEG38PP.

To specify multiple page segments, enter multiple keywords and names separated by blanks. For example: **PAGESEG *SYSTEM** S1IBM ***INLINE** S1LOGO

Environment Notes: Refer to the environment information above.

PRMODE

Specifies job processing. You can use **PRMODE** in one of two ways:

- In MVS, to schedule jobs by specifying installation-defined values
- To specify the type of shift-out and shift-in (SOSI) codes in the data stream. SOSI codes are used to change processing between single-byte and double-byte fonts. SOSI1 and SOSI2 are examples of values that specify the type of SOSI processing.

Validation Rule: **STRING**

Valid Values: 8 (maximum length), for example, SOSI1, SOSI2, and PAGE.

Environment Notes: This option is functionally supported by a printer driver in the MVS, VM, and OS/400 operating systems. In MVS, it is also used as a scheduling parameter. In OS/400, you must be on an IGC machine.

PRTDIRECTION

Ensures page-presentation compatibility between the 3800 Printing Subsystem and the IBM 3835 Page Printer. Use this option when building a form definition (with the ***BUILD** keyword) for print jobs that meet one of the following conditions:

- Were formatted for landscape presentation on wide leading edge media for printing on the 3835
- Were formatted for the 3800 but are now printed on the 3835
- Are printed on both printers.

When printing on the 3835, the print direction specified in the page definition (or to a formatter that produces an AFPDS data stream) should correspond to the **PRTDIRECTION** value specified to PrintManager as follows:

- If you specify PORTRAIT or LANDSCAPE for **PRTDIRECTION**, specify the ACROSS print direction in the page definition or to the formatter.
- If you specify PORTRAIT90 or LANDSCAPE90 for **PRTDIRECTION**, specify the DOWN print direction in the page definition or to the formatter.

Note: Output printed in "portrait" page presentation has the shorter edges of the paper at the top and bottom of the page and the longer edges at the sides of

the page. Output printed in "landscape" page presentation has the longer edges of the paper at the top and bottom of the page and the shorter edges at the sides of the page.

Validation Rule: **LIST**

Valid Values:

PORTRAIT	Portrait page presentation for jobs printed on narrow leading edge media.
PORTRAIT90	Portrait page presentation for jobs printed on wide leading edge media.
LANDSCAPE	Landscape page presentation for jobs printed on wide leading edge media.
LANDSCAPE90	Landscape page presentation for jobs printed on narrow leading edge media.

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

PRTENVIRONMENT Defines the operating system environment of the printer.

Validation Rule: **LIST**

Valid values:

MVS	MVS environment
VM	VM environment
AS/400	AS/400 environment
OS/2	OS/2 environment

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

PRTQUAL Specifies the print-quality level for the print job. Specifying 1 indicates the lowest print quality, and specifying 254 indicates the highest print quality. Actual print-quality levels are unique for each printer; refer to your printer documentation. Values from this option are only used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 254 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

PRTY Specifies the print-job priority. The effect of this option is installation dependent.

Validation Rule: **RANGE**

Valid Values: 0 to 255, where 0 specifies the lowest priority and 255 the highest priority.

Environment Notes: This option is functionally supported in MVS and OS/400.

RSCSID Specifies the userid of the local RSCS service machine in VM. **RSCSID** is used with the **OUTPUTID** option, and **RSCSID** is required when the user is on a different network node from the VM or MVS node where the job will be processed.

Validation Rule: **STRING**

Valid Values: 8 (maximum length)

Environment Notes: This option is functionally supported in the VM environment, and it corresponds to the userid specified on the *TO* parameter of the CP SPOOL command. If the *nodeid.outputid* format is specified on the **OUTPUTID** option in VM, **RSCSID** should also be specified. Otherwise, the user's virtual printer device (00E) must be spooled to the userid of the local RSCS service machine before running the print application, or the user must transfer the print job to the userid of the local RSCS service machine after running the print application. This value correlates to the *TO* parameter on the CP SPOOL command. This value indicates the userid under which printed or punched output is produced.

In VM, **RSCSID** and **DOCOWNER** are mutually exclusive. If both are specified, the **DOCOWNER** value will be used. If the print job must be directed to a remote system using **RSCSID**, **DOCOWNER** must not be specified.

SCHEDULE Specifies when the spooled output file is available to a printer driver.

Validation Rule: **LIST**

Valid Values:

JOBEND	File is available when processing is completed for the job containing the file.
FILEEND	File is available when processing is completed for the file.
IMMED	File is available immediately.

Environment Notes: This option is functionally supported by a printer driver in the OS/400 operating system.

TECHNOLOGY Identifies the type of technology of the specified device.

Validation Rule: **LIST**

Valid Values:

UNKNOWN	Undefined device
VECTORPLOTTER	Vector plotter
RASTERDISPLAY	Raster display
RASTERPRINTER	Raster printer
IMPACTPRINTER	Impact printer
POSTSCRIPT	PostScript device

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

TEXTSET Specifies the MO:DCA-P text capability supported by the printer driver.

Validation Rule: **LIST**

Valid Values:

NONE	No text support

	PT1	PTOCA PT1 text supported
	PT2	PTOCA PT2 text supported.

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

TOPMAR

Specifies the width of the top margin in pels. Values from this option are only used to build an inline form definition when the ***BUILD** keyword is specified for the **FORMDEF** option.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 0 (minimum), 32767 (maximum)

Environment Notes: This option is functionally supported by a printer driver in the OS/400, MVS, and VM operating systems.

TRC

Specifies if the print data contains font table reference characters (TRCs). Fonts referenced by TRCs can be named in the **FONT** option. This option is valid if the data stream is line or mixed, the data stream contains TRCs, and the job will be printed on a 3800-1 or on an AFP printer.

Validation Rule: **LIST**

Valid Values:

YES	Print data contains TRCs.
NO	Print data does not contain TRCs.

Environment Notes: This option is functionally supported by a printer driver in the MVS, VM, and OS/400 operating systems.

UCS

Specifies the following:

- The universal character set image to use in printing the file
- A print train to use in printing the file on an impact printer
- A character-arrangement table for a file printed on an IBM 3800 line printer on a JES2 system.

Validation Rule: **STRING**

Valid Values: 4 (maximum length)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system.

VERTRES

Specifies the vertical resolution of the printer in pels per meter.

Validation Rule: **RANGE**

Valid Values: 0 (decimal places), 1 (minimum), 2^{31} (maximum)

Environment Notes: This option can only be set and queried. It is used to describe the capabilities of a print system component.

WRITER

Specifies a spool program other than JES to process the file.

Validation Rule: **STRING**

Valid Values: 8 (maximum length)

Environment Notes: This option is functionally supported by a printer driver in the MVS operating system.

PrintManager Print Options and Existing Operating System Methods

Table A-1 lists the PrintManager print options and commonly used existing methods of specifying print job attributes in the VM, MVS, and OS/400 operating systems. For more information refer to "Related Documentation" on page 1-5.

Table A-1 (Page 1 of 3). PrintManager Print Options and Existing Operating System Methods

PrintManager Option	MVS	VM	OS/400
CC	RECFM = *xx*A or *xx*M where *xx* can equal FB, VB,F, V, FT, and so forth, in the DD JCL statement.	CC or NOCC options in the PRINT or PSF commands.	CTLCHAR in OVRPRTF command; CC NJE input parameter.
CKPTLINE	CKPTLINE in OUTPUT JCL statement.		
CKPTPAGE	CKPTPAGE in OUTPUT JCL statement.	CKPTPAGE in PSF command.	
CKPTSEC	CKPTSEC in OUTPUT JCL statement.		
CLASS	CLASS in OUTPUT JCL statement or SYSOUT in DD JCL statement.	CLASS in SPOOL PRT command.	
COPIES	COPIES in OUTPUT and DD JCL statements.	COPY in SPOOL PRT command.	COPIES in CHGSPLFA command.
DATACK	DATACK in OUTPUT JCL statement.	DATACK in PSF command.	DATACK NJE input parameter.
DATATYPE			DEVTYPE in OVRPRTF command; PRMODE NJE input parameter.
DOCOWNER		SPOOL 00E FOR *userid*	
DUPLEX	Specified in form definition.	DUPLEX in PSF command	DUPLEX in OVRPRTF command; DUPLEX NJE input parameter.
FCB	FCB in DD JCL statement or PDEF in OUTPUT JCL statement.	FCB in SPOOL PRT command.	

Table A-1 (Page 2 of 3). PrintManager Print Options and Existing Operating System Methods

PrintManager Option	MVS	VM	OS/400
FIDELITY			FIDELITY in CHGSPLFA command.
FLASHCNT	FLASH in OUTPUT or DD JCL statements.	FLASH in SPOOL PRT command.	
FLASHNAME	FLASH in OUTPUT or DD JCL statements.	FLASH in SPOOL PRT command.	
FORM	FORMS in OUTPUT JCL statement or SYSOUT in DD JCL statement.	FORM in SPOOL PRT command.	FORMTYPE in CHGSPLFA command.
INBIN	Specified in form definition.	BIN in the PSF command	DRAWER in OVRPRTF command; BIN NJE input parameter.
INDEX	Supported in OUTPUT JCL statement (3211 printer only).		
LINDEX	Supported in OUTPUT JCL statement (3211 printer only).		
LINECT	LINECT in OUTPUT JCL statement (line printer only).		
MESSAGES	PIMSG in OUTPUT JCL statement.	MESSAGE RETURN in PSF command.	
MODIFYNAME	MODIFY under JES only.	MODIFY in SPOOL PRT command.	
MODIFYTRC	MODIFY under JES only.	MODIFY in SPOOL PRT command.	
OUTDISP	HOLD option in DD JCL statement.	HOLD or NOHOLD options in SPOOL PRT command.	HOLD and SAVE in OVRPRTF command.
OUTMETHOD = BURST	BURST = Y,N in OUTPUT or DD JCL statements.		

Table A-1 (Page 3 of 3). PrintManager Print Options and Existing Operating System Methods

PrintManager Option	MVS	VM	OS/400
OUTPUTID = *nodeid.outputid*	DEST = *nodeid.outputid* in OUTPUT JCL statement.	TAG DEV 00E *nodeid* and SPOOL 00E DEST *outputid*.	
OUTPUTID = *outputid*	DEST = *outputid* in OUTPUT JCL statement.	SPOOL 00E DEST *outputid*.	DEV in CHGSPLFA command.
OUTQ			OUTQ in CHGSPLFA command.
PAGERANGE			PAGERANGE in CHGSPLFA command.
PRMODE	PRMODE in OUTPUT JCL statement.	PRMODE in PSF command.	IGCDTA and IGCSOSI in OVRPRTF command.
PRTY	PRTY in OUTPUT JCL statement under JES only.		OUTPTY in CHGSPLFA command.
RSCSID		SPOOL 00E TO *rscs userid*.	
SCHEDULE			SCHEDULE in CHGSPLFA command.
TRC	TRC in OUTPUT JCL statement or DCB in DD JCL statement.	TRC or NOTRC options in the PRINT or PSF commands.	TRC NJE input parameter.
UCS	UCS in OUTPUT or DD JCL statements.		
WRITER	Supported in OUTPUT JCL statement under JES only.		

Appendix B. PrintManager C Applications

This appendix provides the following information about PrintManager C applications:

- C language coding conventions
- Creating and running PrintManager C applications in MVS and VM
- Compiling and running PrintManager Interface C applications in OS/400
- Using PrintManager error verbs
- PrintManager Interface data types for the C language
- PrintManager Interface C language verb reference
- C language example.

C Language Coding Conventions

Verb Calls

In C language, you can use the PrintManager verb long names or short names when coding an application. For example, when calling the Abort verb, you can use SPRABRT (Abort) or SPRABRT. The coding example shown in "Example of a PrintManager Session in C Language" on page B-31 uses the long names for the verbs.

Header Files

The PrintManager header files are shipped with the IBM C/400 compiler (for OS/400) or with PrintManager (for MVS and VM). The application source program must have two define statements (for the PrintManager Interface verbs and the PrintManager Interface error codes) followed by an include statement for the system header file. These statements must be coded before any calls to PrintManager Interface verbs. These statements are as follows:

```
#define  INCL_PRTMGR
#define  INCL_ERRORS
#include  <ekipmgr.h>
```

If your PrintManager Interface application makes calls to the API verbs, you must also code:

```
#define  INCL_PRD
```

APIENTRY

All functions are implicitly of type APIENTRY. The entry is defined in the system header file EKIPMGR.H as:

```
#define APIENTRY
```

Buffers

In C language, when a buffer contains a list of names (for example, with the SPRLOPT), this is implemented with an array of pointers that refer to the list in the buffer. Otherwise, the buffer contains an array of structures, or data.

When allocating storage for a structure (except for the SDF structure), the *sizeof* C function should be used to determine the size of the structure because the C compiler may add additional bytes for structure alignment or padding between fields of a structure.

Creating and Running PrintManager C Applications in MVS and VM Environment

In the MVS and VM environments, PrintManager provides dynamic linking of PrintManager C language applications to the PrintManager program code. Dynamic linking allows changes to be made to the PrintManager licensed program without requiring you to modify or relink your PrintManager API and PrintManager Interface applications. Furthermore, dynamic linking provides for smaller application executable modules because PrintManager applications will only link to verb "stub" libraries at link-edit time. Using stub libraries, however, requires that the PrintManager execution time libraries are available when running a PrintManager application.

Note: PrintManager in MVS/ESA, VM/XA, or VM/ESA systems only supports IBM SAA PrintManager applications operating in 31 bit addressing mode. Applications in these environments must specify the AMODE 31 option at program link-edit or load time. On VM/SP Release 6 systems, specifying AMODE 31 is recommended but not required.

For general information on how to create and run C programs in MVS and VM, refer to *IBM C/370 User's Guide*, SC09-1264. The following sections describe PrintManager-specific information for compiling, link-editing, and running C PrintManager applications in MVS and VM environments.

Creating and Running MVS PrintManager C Applications

The following sections tell how to compile, link-edit, and run C applications in the MVS environment.

Note: The library names shown in the examples in the following sections are examples only. For the actual names of these libraries, refer to your program directory.

Compiling MVS Applications

As described in "Header Files" on page B-1, any application source file making calls to PrintManager verbs must include the PrintManager header files. These header files are installed in a PrintManager header file library 'EKI.V*v*R*r*M*m*.EKIHSRC'. This library must be concatenated to the SYSLIB DD statement of your compile job, along with the C/370 product system header file library 'EDC.V*v*R*r*M*m*.SEDCHDRS'.

Note: For both 'EKI.V*v*R*r*M*m*.EKIHSRC' and 'EDC.V*v*R*r*M*m*.SEDCHDRS', replace *v*, *r*, and *m* respectively with the version, release, and modification of PrintManager and C/370 that you have installed.

Link-Editing MVS Applications

After you successfully compile your application modules, they must be link-edited, along with the PrintManager and C/PLI stub libraries. Figure B-1 shows an example of an MVS batch JCL job for link-editing PrintManager applications.

```
//JOBname     JOB    acctno,name,...
//stepname    EXEC   PGM=IEWL,PARM='AMODE=31,RMODE=ANY,..'
//SYSPRINT    DD     SYSOUT=A
//SYSLMOD     DD     DSNAME=datasetname(member),UNIT=SYSDA,
                     DISP=(NEW,PASS),SPACE=(subparams)
//SYSLIB      DD     DSNAME=EDC.V1R2M0.SEDCBASE,DISP=SHR
                     DSNAME=PLI.V2R2M1.SIBMBASE,DISP=SHR
                     DSNAME=EKI.V1R1M0.EKIPMLIB,DISP=SHR
//SYSLIN      DD     DSNAME=datasetname(member),DISP=SHR
//SYSUT1      DD     UNIT=SYSDA,SPACE=(subparams)
/*
```

Figure B-1. Example MVS Batch Job for Link-Editing PrintManager Applications

Running MVS Applications

Before running an MVS PrintManager application, you must make the PrintManager, C, and PLI execution time libraries accessible to the application. You can do this by either:

- Installing the PrintManager execution time library in the Link Pack Area (LPA). In this case, do not include the execution time library in the STEPLIB DD statement.
- Explicitly concatenating the libraries to the STEPLIB DD statement at the program execution step. In this case, the libraries are *not* installed in the LPA.

Figure B-2 shows an example of running an MVS application with the library concatenation at the program execution step because the PrintManager, C, and PLI libraries are not installed in the LPA.

```
//JOBname     JOB    acctno,name,...
//STEP1       EXEC   PGM=progname,PARMS='parms'
//STEPLIB     DD     DSN=EDC.V1R2M0.SEDCLINK,DISP=SHR
//            DD     DSN=PLI.V2R2M1.SIBMLINK,DISP=SHR
//            DD     DSN=EKI.V1R1M0.EKIRUNLB,DISP=SHR
//EKIGLNME    DD     DSN=EKI.$GLDS$.EKIPM,DISP=SHR
/*
```

Figure B-2. Example of Running an MVS PrintManager Application

If the PrintManager execution time library cannot be found, the SPRINIT (Initialize PrintManager) verb will return a null *hab* parameter, and dynamic linking will not occur.

Creating and Running VM PrintManager C Applications

The following sections tell how to compile, link-edit, and run C applications in the VM environment.

Compiling VM Applications

As described in "Header Files" on page B-1, ensure that the PrintManager header files are accessible to the C/370 compiler on your VM system.

Link-Editing VM Applications

When a PrintManager application is link-edited or loaded, the PrintManager stub library is a txtlib that must be included in the txtlib concatenation at link-edit or load time, along with the C/PLI stub library. Figure B-3 shows an example of the CMS command sequence you can use to create an executable module **pmappl** from two C program files **appmain** and **appsub**.

```
GLOBAL TXTLIB EDCBASE IBMLIB CMSLIB EKIPMLIB
LOAD appmain appsub (RESET CEESTART
GENMOD pmappl (FROM CEESTART
```

Figure B-3. Example CMS Command Sequence for Link-Editing PrintManager Applications

The CMS LKED command may also be used to create executable PrintManager applications in a CMS LOADLIB. Refer to the publications listed in "For VM" on page 1-7 for more information on CMS commands.

Running VM Applications

Before running a VM PrintManager application, you must make the PrintManager and C/PLI common execution time libraries accessible to the application. If the PrintManager execution time library is not installed as a VM Discontiguous Shared Segment (DCSS), do one of the following:

- Load the PrintManager execution time library into your own nucleus extension space, using the following CMS commands:

  ```
  FILEDEF ekild DISK EKIRUNLB LOADLIB *
  NUCXLOAD EKIDISPT EKIDISPT ekild (SYSTEM
  ```

- Add the PrintManager execution time load library EKIRUNLB into your global load library list using the CMS GLOBAL LOADLIB command.

If the PrintManager execution time library cannot be found or loaded, the SPRINIT (Initialize PrintManager) verb will return a null *hab* parameter, and dynamic linking will not occur.

Note: On VM/SP Release 5, if the execution time library cannot be found or loaded, a CMS abend will occur, which indicates that entry point EKIDISPT is undefined or unresolved.

Invalid Handles

In MVS and VM, when you pass an invalid handle (in the *hab*, *hprm*, or *hprd* parameters) to a PrintManager verb, your application program will abend with abend code X'0245'. A program dump is initiated unless the following C pragma preprocessor directive is coded in your main C routine:

```
#pragma runopts(nospie, nostae)
```

This pragma shown above disables the C/PLI abend handler. Refer to *IBM C/370 User's Guide* for more information on the C pragma directive.

Compiling and Running PrintManager C Applications in the OS/400 Environment

Compiling and running PrintManager Interface C applications in the OS/400 Environment requires the following:

1. Use the PrintManager header files as described in "Header Files" on page B-1.
2. After compiling your application, the SETPGMINF command must be executed to set program environment information. The library information file which must be included in this command is QACJINFO located in the QSYS library. The following is an example of how to use the SETPGMINF command for a user's program named MYPGM in the library named MYLIB:

   ```
   SETPGMINF ROOTPGM(MYLIB/MYPGM) LIBFILE((QSYS/QACJINFO) (*CLIB))
   ```

   For more information about the SETPGMINF and EXTPGMINF commands, refer to *Application System/400 Languages: C/400 User's Guide*, SC09-1303.

Using PrintManager Error Verbs

Except for the SPRINIT and SPRTERM verbs, if a PrintManager verb returns a FALSE or 0 return value, you can use the PrintManager error verbs to obtain information about the error as follows:

- Use the SPRGERI verb to get the error-information structure (**ERRINFO**) associated with the last PrintManager Interface error. The error-information structure contains the error id. The error id is a four byte value in which the first two bytes are the severity of the error and the last two bytes are the error code.

  A severity code of 4 indicates a warning condition, while a severity code of 8 indicates an error condition. A warning condition means that the requested function completed successfully, and you may continue, although results may not be as expected. For example, an error of X'40B1' (16561) has a severity code of 4, and indicates that truncation occurred when a valid value was set in a print descriptor. An error condition means that an error occurred, and the requested function did not complete successfully.

- For some errors, additional error information can be obtained by using the error-information structure (**ERRINFO**) as input to the SPRGEEM verb. The SPRGEEM verb returns additional error information in the **ERRMSG** structure. The **ERRMSG** structure contains the message identifier and a value which is a string containing additional error information.

  Note: Appendix F, Verb Error Codes lists PrintManager error codes and provides general explanations of associated errors, a list of PrintManager Interface verbs that may have issued the error, user responses, and a list of additional messages (if any) that are issued.

- After the error-information structure (**ERRINFO**) from SPRGERI is no longer needed, use the SPRFERI verb to free storage obtained for this structure.

Data Types for C Language

This section describes the data types for C language, which are similar to the general data types described in "PrintManager Interface Verb Data Types" on page 4-2. Each description of a C data type, therefore, provides data characteristics or mappings that are unique to the C language.

Data Type	Description
BOOL	In C, TRUE is defined as a nonzero value, and FALSE is defined as 0. Do not use the constant TRUE in an equality comparison with another BOOL data type, because this may produce incorrect results. Takes the format: `typedef unsigned short BOOL;`
BYTE	A byte of data in the following format: `typedef unsigned char BYTE;`
CHAR	A single-byte character in the following format: `typedef unsigned char CHAR;`

ERRINFO	The data structure used with the SPRGERI verb to return error information in the following format:
	`typedef struct _ERRINFO`
	`{`
	`USHORT cbFixedErrInfo;`
	`ERRORID idError;`
	`USHORT cDetailLevel;`
	`USHORT offaoffszMsg;`
	`USHORT offBinaryData;`
	`} ERRINFO;`

ERRMSG	The data structure used by the &errm. verb to return additional error information in the following format:
	`typedef struct _ERRMSG`
	`{`
	`ULONG idIdentifier;`
	`CHAR szValue[256];`
	`} ERRMSG;`

ERRORID	A four-byte value in which the first two bytes are the severity and the last two bytes are the error code in the following format: `typedef ULONG ERRORID:`
HAB	Handle to a PrintManager anchor block in the following format: `typedef LHANDLE HAB;`
HPRM	A print-session identifier in the following format: `typedef LHANDLE HPRM;`
LHANDLE	Pointer for the handle for PrintManager verbs in the following format: `typedef void *LHANDLE`
LONG	A signed four-byte integer value (32 bits) in the following format:

```
typedef long LONG;
```

OPTDATA1 A print-option data structure in the following format:

```
typedef struct optdata1
   {
   PSZ     pszOption;
   ULONG   flValType;
   ULONG   cbValLength;
   PBYTE   pbValue;
   } OPTDATA1;
```

PBYTE Pointer to a byte of data in the following format:

```
typedef BYTE  *PBYTE;
```

PERRINFO Pointer to an ERRINFO structure in the following format:

```
typedef ERRINFO  *PERRINFO;
```

PERRMSG Pointer to an ERMSG structure in the following format:

```
typedef ERRMSG  *PERRMSG;
```

PSDF Pointer to an SDF structure in the following format:

```
typedef SDF  *PSDF;
```

PSZ Pointer to a zero-terminated string in the following format:

```
typedef char  *PSZ;
```

PSZARRAY Pointer to an array of zero-terminated strings in the following format:

```
typedef PSZ  *PSZARRAY
```

PULONG Pointer to a ULONG data type in the following format:

```
typedef ULONG  *PULONG;
```

SDF A self-defining field (structure) in the following format:

```
typedef struct _SDF {
   ULONG  cbLength;
   ULONG  flType;
   BYTE   bData[1];
   } SDF;
```

ULONG An unsigned four-byte integer value (32 bits) in the following format:

```
typedef unsigned long ULONG;
```

USHORT An unsigned two-byte integer value (16 bits) in the following format:

```
typedef unsigned short USHORT;
```

SPRABRT (Abort)
PrtMgrAbort

MVS	VM	OS/400	OS/2
X	X	X	

Function

Aborts the print session.

Syntax

```
extern BOOL APIENTRY SPRABRT   (HPRM hprm);
```

Figure B-4. C Syntax of the SPRABRT Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

Usage

Use the SPRABRT verb to abort the print session and to delete the print job (if active). When the SPRABRT verb is issued, the *hprm* from the aborted session is no longer valid.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Ending your application without calling SPRCLOS or SPRABRT could cause unpredictable results.

Return Values

***success* (BOOL)**
A nonzero value means the print session was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADOC (Abort Document) PrtMgrAbortDoc

MVS	VM	OS/400	OS/2
X	X	X	

Function

Aborts and deletes the print job.

Syntax

```
extern BOOL APIENTRY SPRADOC   (HPRM hprm);
```

Figure B-5. C Syntax of the SPRADOC Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

Usage

Use the SPRADOC verb to abort processing of the print job and to delete it. When the SPRADOC verb is issued, PrintManager returns to Session Active state.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

SPRADOC does not abort a print job that has already been ended with the SPREDOC verb.

Return Values

***success* (BOOL)**
A nonzero value means the print job was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADDF (Add File)
PrtMgrAddFile

MVS	VM	OS/400	OS/2
X	X	X	

Function

Writes a file of print data to the print job.

Syntax

```
extern BOOL APIENTRY SPRADDF   (HPRM hprm,
                                PSZ pszFileName);
```

Figure B-6. C Syntax of the SPRADDF Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

PSZ pszFileName
(Input). A string containing the name of a file with print data to be sent to the spool. The file name follows the system-specific naming convention for each environment:

VM	filename filetype [filemode] or filename.filetype[.filemode]
MVS	QUAL1.QUAL2.QUAL3.QUAL4.QUAL5
OS/400	library/file(member)

Usage

Use the SPRADDF verb to write a file of print data to the print job.

If an error occurs, PrintManager returns to Session Active state with the exception of two errors: 16412 ("cannot open file") and 16547 ("read error"). These two errors will leave you in Job-in-Progress state. See "PrintManager Interface Operating States" on page 3-5 for more information on operating states.

Note: In OS/400, the file specified in the *FileName* parameter must be a physical data file with FILETYPE (*DATA) and LVLCHK (*NO).

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**

A nonzero value means the named file was written to the print job, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRCLOS (Close)
PrtMgrClose

MVS	VM	OS/400	OS/2
X	X	X	

Function

Ends the current print session.

Syntax

```
extern BOOL APIENTRY SPRCLOS   (HPRM hprm);
```

Figure B-7. C Syntax of the SPRCLOS Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

Usage

Use the SPRCLOS verb to end a print session and to free storage associated with the session. When SPRCLOS is issued, the *hprm* parameter from the closed session is no longer valid.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Ending your application without calling SPRCLOS or SPRABRT could cause unpredictable results.

Return Values

***success* (BOOL)**
A nonzero value means the print session ended successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPREDOC (End Document)
PrtMgrEndDoc

MVS	VM	OS/400	OS/2
X	X	X	

Function

Ends the print job.

Syntax

```
extern USHORT APIENTRY SPREDOC   (HPRM hprm);
```

Figure B-8. C Syntax of the SPREDOC Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

Usage

Use SPREDOC to end a print job. When the SPREDOC verb is issued, PrintManager returns to Session Active state.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (USHORT)**
A nonzero value means the print job ended successfully, whereas a value of 0 means an error occurred. In VM, the non-zero return value is the print job ID.

Environment Restrictions

None.

SPRFERI (Free Error Information) PrtMgrFreeErrorInfo

MVS	VM	OS/400	OS/2
X	X	X	

Function

Frees storage for the error-information structure returned from the SPRGERI verb.

Syntax

```
extern BOOL APIENTRY SPRFERI   (PERRINFO pErrorInfo);
```

Figure B-9. C Syntax of the SPRFERI Verb

Parameters

PERRINFO pErrorInfo
(Input). The **ERRINFO** structure.

Usage

Use SPRFERI to free storage for the error-information structure you specify in *ErrorInfo*. You should issue SPRFERI for every error-information structure returned with the SPRGERI verb.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

***success* (BOOL)**
A nonzero value means storage was freed successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGEEM (Get Error Message) PrtMgrGetExtErrorMsg

MVS	VM	OS/400	OS/2
X	X	X	

Function

Gets additional error information associated with the error-information structure.

Syntax

```
extern BOOL APIENTRY  SPRGEEM   (PERRINFO pErrorInfo,
                                 ULONG ulIndex,
                                 PERRMSG pErrorMsg,
                                 PULONG pulTotalCount);
```

Figure B-10. C Syntax of the SPRGEEM Verb

Parameters

PERRINFO pErrorInfo
(Input). The error-information structure (**ERRINFO**) returned by SPRGERI.

ULONG ulIndex
(Input). Specifies which additional message to return. Use a value of 1 to select the first message, a value of 2 to select the second message, and so on.

PERRMSG pErrorMsg
(Output). The buffer containing the **ERRMSG** structure.

PULONG pulTotalCount
(Output). SPRGEEM updates this field to specify the number of additional messages associated with the error-information structure.

Usage

The SPRGEEM verb returns additional error information in the **ERRMSG** structure. The original **ERRINFO** structure returned from SPRGERI must be passed to SPRGEEM. It must not be a copy.

Information in the **ERRMSG** structure is often useful in further resolving the cause of errors. For example, for error code PMERR_PRM_CANNOT_OPEN_FILE, the *value* field of the **ERRMSG** structure provides the name of the file where the error occurred. For operating system errors, PrintManager updates the *value* field with error text provided by the operating system. Information in the *value* field is truncated to 256 characters, if necessary.

There may be 1 to *n* additional error messages associated with the **ERRINFO** structure. Set *Index* to 0 to get the total number of messages that can be returned. If the value returned in *TotalCount* is 0, there are no additional messages. If there are additional messages, set *Index* to correspond to the message you want to return. If you want to return all additional messages, call SPRGEEM repeatedly, incrementing *Index* until *TotalCount* is reached.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

***success* (BOOL)**

A nonzero value means that the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGERI (Get Error Information) PrtMgrGetErrorInfo

MVS	VM	OS/400	OS/2
X	X	X	

Function

Gets error-information structure from the last error.

Syntax

```
extern PERRINFO APIENTRY SPRGERI   (HAB hab);
```

Figure B-11. C Syntax of the SPRGERI Verb

Parameters

HAB hab
(Input). The anchor-block handle returned from a successful call to SPRINIT. See Appendix E, "System-Specific Information" on page E-1 for more information on SPRINIT.

Usage

Use the SPRGERI verb to get the error-information structure associated with the last PrintManager Interface error. The error information structure contains the error id. The error id is a four byte value in which the first two bytes are the severity and the last two bytes are the error code. You can then use the error-information structure (**ERRINFO**) as input to the SPRGEEM verb to get additional information (if any) about the error. To free storage for the error-information structure, use SPRFERI.

For the *hab* parameter, ensure that you enter the anchor-block handle returned by the SPRINIT (Initialize PrintManager) verb. Otherwise, in the 370 environment PrintManager will abend with an abend code of X'0245' and in OS/400 a value of 0 will be returned.

Notes:

1. If PrintManager cannot be initialized (with SPRINIT (Initialize PrintManager)), SPRGERI (Get Error Information) will not return an error-information buffer.
2. Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

***ERRINFO* (ERRINFO)**

The SPRGERI verb returns an **ERRINFO** structure for the last error. See Appendix F, "Verb Error Codes" on page F-1 for information on PrintManager Interface verb error codes. A return value of 0 indicates that no previous error occurred.

As well as errors from PrintManager Interface verbs, you can get errors from API verbs because some PrintManager Interface verbs invoke PrintManager API verbs.

Environment Restrictions

None.

SPRINIT (Initialize PrintManager)
PrtMgrInitialize

MVS	VM	OS/400	OS/2
X	X	X	

Function

Initialize PrintManager.

Syntax

```
extern HAB APIENTRY SPRINIT    (USHORT fsOptions);
```

Figure B-12. C Syntax of the SPRINIT Verb

Parameters

USHORT fsOptions
(Input). This parameter is reserved and should have a value of 0.

Usage

Use the SPRINIT verb to initialize PrintManager. You cannot issue any other API verbs (including error verbs) until you have successfully initialized PrintManager.

Return Values

***HAB* (HAB)**
The SPRINIT verb should return an anchor-block handle (HAB). A return value of 0, however, indicates an error while initializing PrintManager.

Ensure that you save the HAB value that the SPRINIT verb returns. You will need to specify this value on other PrintManager verbs. You cannot issue two successive calls to SPRINIT. You must terminate PrintManager with the SPRTERM verb before reinitializing PrintManager.

Environment Restrictions

None.

SPRLOPT (List Options)
PrtMgrListOptions

MVS	VM	OS/400	OS/2
X	X	X	

Function

List current print options for a print session.

Syntax

```
extern BOOL APIENTRY SPRLOPT   (HPRM hprm,
                                ULONG ulLevel,
                                ULONG cbLength,
                                PBYTE pbBuffer,
                                PULONG pulLengthNeeded,
                                PULONG pulItemsReturned,
                                PULONG pulItemsRemaining,
                                PSDF psdfContinue);
```

Figure B-13. C Syntax of the SPRLOPT Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

ULONG ulLevel
(Input). Specifies the format of the data in the buffer. A value of 1 specifies that a list of print-option names is returned in the buffer as an array of character strings.

ULONG cbLength
(Input). Specifies the buffer size (see "Usage").

PBYTE pbBuffer
(Output). Returns a list of current print-option names. If you did not specify sufficient buffer space as indicated on the *Length* parameter, *Buffer* contains a partial list, and *LengthNeeded* indicates a value for the additional space needed to complete the list.

PULONG pulLengthNeeded
(Output). If the buffer is large enough to contain all option names in the requested list, *LengthNeeded* indicates the actual size of the list returned. If not, *LengthNeeded* indicates a value for the buffer space needed to complete the list.

PULONG pulItemsReturned
(Output). Indicates the number of print-option names returned in the buffer.

PULONG pulItemsRemaining
(Output). For a partial list, indicates the number of option names not yet listed; otherwise, *ItemsRemaining* is 0.

PSDF psdfContinue
(Input/Output). Used to get a list of options in a series of calls (see "Usage"). If you are not using this field for a series of calls, specify it as **NULL**.

Usage

Use the SPRLOPT verb to list the set of current options in a print session. If you do not know the buffer size required for the list, do one of the following:

- If storage is not constrained or if the list is small, issue SPRLOPT (List Options) in two calls. First, specify *Length* as 0 and *Buffer* as **NULL**. *LengthNeeded* will indicate a value for the buffer space needed for the list.

  For the second call, allocate the buffer space required (as returned in *LengthNeeded* on the first call) and specify the required buffer space in *Length*. The entire list will be returned in the buffer.

- If storage is constrained or if the list is large, issue SPRLOPT (List Options) in a series of calls. First, allocate sufficient buffer space (at least enough to contain one print descriptor name in the list). For the first call specify *Length* as the amount of storage available in the buffer you are using, and allocate storage for the *Continue* parameter equal to the value of the ***PRTMGR_GETFIRST_LENGTH*** constant. When allocating storage, do *not* use the *sizeof* C function on the SDF structure to determine the size of the parameter. In the parameter set the *Type* field to ***PRTMGR_GET_FIRST*** and the *Length* field to ***PRTMGR_GETFIRST_LENGTH***. In the *Continue* parameter set the *Type* field to ***PRTMGR_GET_FIRST*** and the *Length* field to ***PRTMGR_GETFIRST_LENGTH***.

  Continue the series of calls (and continue the list) by passing back the *Continue* parameter unmodified. When *ItemsRemaining* is 0, you will have received all list entries.

Note: Any changes to the print options during a series of calls using *Continue* will not be reflected in the partial lists returned.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means the list request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPROPEN (Open)
PrtMgrOpen

MVS	VM	OS/400	OS/2
X	X	X	

Function

Opens and identifies a print session.

Syntax

```
extern HPRM APIENTRY SPROPEN   (HAB hab,
                                PSZ pszPrdName,
                                ULONG cbLength,
                                PBYTE pbBuffer);
```

Figure B-14. C Syntax of the SPROPEN Verb

Parameters

HAB hab
(Input). The anchor-block handle returned from a successful call to SPRINIT. See Appendix E, "System-Specific Information" on page E-1 for more information on SPRINIT.

PSZ pszPrdName
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored if it is **NULL**, a zero-length string, all blanks, or a single asterisk (*), and no validation is done.

For information on naming conventions and merging, refer to *PrintManager Application Programming Interface Reference*.

ULONG cbLength
(Input). This field is reserved and must have a value of zero.

PBYTE pbBuffer
(Input). This field is reserved and must have a value of **NULL**.

Usage

Use the SPROPEN verb to start a print session. You can also use SPROPEN to define the set of current options by specifying a print descriptor.

Return Values

***hprm* (HPRM)**

The SPROPEN verb should return the session identifier (*hprm*). A return value of 0, however, means an error occurred. Ensure that you save the *hprm* value that the SPROPEN verb returns. You will need to specify this value on other PrintManager Interface verbs used in a session.

For the *hab* parameter, ensure that you enter the anchor-block handle returned by the SPRINIT (Initialize PrintManager) verb. Otherwise, in the 370 environment PrintManager will abend with an abend code of X'0245' and in OS/400 a value of 0 will be returned.

Environment Restrictions

In VM, only one print session can be open at a time. In all other systems, multiple sessions can run concurrently. (See Appendix E, "System-Specific Information" on page E-1 for more information.)

SPRQOPT (Query Option)
PrtMgrQueryOption

MVS	VM	OS/400	OS/2
X	X	X	

Function

List current values for a print option.

Syntax

```
extern BOOL APIENTRY SPRQOPT   (HPRM hprm,
                                ULONG ulLevel,
                                ULONG cbLength,
                                PBYTE pbBuffer,
                                PULONG pulLengthNeeded);
```

Figure B-15. C Syntax of the SPRQOPT Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

ULONG ulLevel
(Input). Specifies the format of the print-option information passed and returned in the *Buffer* parameter. On input, specify the print-option name, which can be up to 31 characters, in the *Buffer* parameter. Any additional characters will be truncated. On output, the format of the *Buffer* fields depend on the level you specify as follows:

1 Returns an **OPTDATA1** structure with values for the *Option*, *ValType*, *ValLength*, and *Value* fields.

ULONG cbLength
(Input). Specifies the buffer size (see "Usage").

PBYTE pbBuffer
(Input/Output). Returns a print-option information structure as specified in the *Level* parameter. If you did not specify sufficient buffer space as indicated on the *Length* parameter, *Buffer* does not contain any information, and *LengthNeeded* indicates a value for the space needed.

Information passed in the **OPTDATA1** structure will be modified when returned.

PULONG pulLengthNeeded
(Output). If the buffer is large enough to contain the print-option information, *LengthNeeded* indicates the actual size of the information returned. If *Length* does not specify sufficient buffer space for the requested information, *LengthNeeded* indicates a value for the buffer space needed for the specified print-option information. If you specify *Length* as 0 and *Buffer* as **NULL**, *LengthNeeded* indicates a value for the buffer size needed for the largest print option in the set of current options according to the value specified in the *Level* parameter.

Usage

Use the SPRQOPT verb to list the current values for a print option. You can issue SPRQOPT in two calls. First, specify *Length* as 0 and *Buffer* as **NULL**. *LengthNeeded* will indicate a value for the buffer size needed for the largest print option.

For the second call, allocate the buffer space required (as returned in *LengthNeeded* on the first call) and indicate the amount of buffer space in *Length*. The option information will be returned in the buffer. Because *LengthNeeded* reflects the size of the buffer needed for the largest print option in the set of current options, this buffer is sufficient for any of the current options.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**

A nonzero value means the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRSDOC (Start Document)
PrtMgrStartDoc

MVS	VM	OS/400	OS/2
X	X	X	

Function

Starts a print job and specifies a document name for the print job.

Syntax

```
extern BOOL APIENTRY SPRSDOC   (HPRM hprm,
                                PSZ pszDocName);
```

Figure B-16. C Syntax of the SPRSDOC Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

PSZ pszDocName
(Input). A string containing the document name for the print job. If no document name is specified, the name is inherited from the previous print job in a session. If no previous document name exists, the operating system provides a default. The *DocName* parameter must follow the system-specific naming conventions.

Usage

Use SPRSDOC to start a print job. You must issue SPREDOC before issuing another SPRSDOC within the same print session.

If the SPRSDOC verb is successful, PrintManager is placed in Job-in-Progress state.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means a new print job was initiated, whereas a value of 0 means an error occurred.

Environment Restrictions

DocName is truncated to 10 characters in OS/400. *DocName* can be an 8 character file name and an 8 character file type (separated by a blank) in VM. *DocName* is not supported in the MVS environment.

SPRSOPT (Set Option)
PrtMgrSetOption

MVS	VM	OS/400	OS/2
X	X	X	

Function

Set or change a print option.

Syntax

```
extern BOOL APIENTRY SPRSOPT   (HPRM hprm,
                                PSZ pszPrdName,
                                ULONG ulLevel,
                                ULONG cbLength,
                                PBYTE pbBuffer);
```

Figure B-17. C Syntax of the SPRSOPT Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

PSZ pszPrdName
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored if it is **NULL**, a zero-length string, all blanks, or a single asterisk (*). If *DEL is specified, all print options are deleted from the set of current options. The print descriptor is no longer in effect, and any options specified in the **OPTDATA1** structure are not validated.

For information on naming conventions and merging, refer to *PrintManager Application Programming Interface Reference*.

ULONG ulLevel
(Input). Specifies the format of the print-option information passed in the buffer. The print-option name can be up to 31 characters. Any additional characters will be truncated. On input, the format of the *Buffer* parameter depends on the level you specify as follows:

1 An **OPTDATA1** structure with values for the *Option*, *ValType*, *ValLength*, and *Value* fields.

If the *Value* field in the **OPTDATA1** structure is **NULL**, a zero-length string, all blanks, or a single asterisk (*), the value in the set of current options is set to the default value from the print descriptor. If no print descriptor has been specified, the option is deleted from the set of current options.

ULONG cbLength
(Input). Specifies the buffer size.

PBYTE pbBuffer
(Input). The print-option information structure as specified in the *Level* parameter.

Usage

Use the SPRSOPT verb to validate and set print-option information and to specify a print descriptor (see "Print Option Validation" on page 5-5).

If an error occurs in retrieving a print descriptor, any defaults and validation information from the previous print descriptor are no longer in effect.

If the *PrdName* parameter is specified as **NULL**, a zero-length string, all blanks, or an asterisk, the options specified on this call are validated against the current print descriptor, if one has been previously defined. If no print descriptor has been defined, no validation is performed; therefore, all of the print options and values specified are valid.

If a print descriptor is specified after print options have been set, the previously set print options are validated against the new print descriptor.

You should specify a print descriptor only once because each time you specify a print descriptor, it is reapplied and the set of current options is validated.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means the print-option values were set successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRTERM (Terminate PrintManager) PrtMgrTerminate

MVS	VM	OS/400	OS/2
X	X	X	

Function

Terminate PrintManager.

Syntax

```
extern BOOL APIENTRY SPRTERM   (HAB hab);
```

Figure B-18. C Syntax of the SPRTERM Verb

Parameters

HAB hab
(Input). The anchor-block handle returned from a successful call to SPRINIT.

Usage

Use the SPRTERM verb to terminate PrintManager and to release all associated resources. For the *hab* parameter, ensure that you enter the anchor-block handle returned on the SPRINIT verb when you initialized PrintManager. You must terminate a PrintManager session before you can reinitialize PrintManager.

For the *hab* parameter, ensure that you enter the anchor-block handle returned by the SPRINIT (Initialize PrintManager) verb. Otherwise, in the 370 environment PrintManager will abend with an abend code of X'0245' and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means PrintManager was terminated successfully, whereas a value of zero means an error occurred.

Environment Restrictions

None.

SPRWRIT (Write)
PrtMgrWrite

MVS	VM	OS/400	OS/2
X	X	X	

Function

Sends a buffer of data to the print job.

Syntax

```
extern BOOL APIENTRY SPRWRIT   (HPRM hprm,
                                LONG lCount,
                                PBYTE pbData);
```

Figure B-19. C Syntax of the SPRWRIT Verb

Parameters

HPRM hprm
(Input). Identifies a valid print session.

LONG lCount
(Input). The length of the buffer of print data sent to the print job.

PBYTE pbData
(Input). The buffer of print data sent to the print job.

Usage

Use the SPRWRIT verb to send a buffer of data to a print job. If an error occurs, PrintManager returns to Session Active state.

For the *hprm* parameter, ensure that you enter a valid print-session identifier. Otherwise, in the MVS and VM environments PrintManager will abend with X'0245', and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means the buffer of data was sent to the print job successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

In VM, for PrintManager Interface applications, the maximum length of the buffer is 32K bytes.

Example of a PrintManager Session in C Language

```
/*********************************************************************/
/*            PRINTMANAGER C LANGUAGE EXAMPLE                        */
/*                                                                   */
/* This example shows how to use PrintManager.                       */
/* It is not intended to satisfy any particular functional needs.    */
/* For example, you typically would not want to create a             */
/* print descriptor for one job and immediately destroy it.          */
/*                                                                   */
/* Sequence of Operations:                                           */
/*                                                                   */
/* - Start a PrintManager process                (PrtMgrInitialize) */
/* - Open a print descriptor session                     (PrdOpen) */
/* - Set rules and valid values for options          (PrdSetOption) */
/* - Save the print descriptor                   (PrdSaveDescriptor) */
/* - Close the print descriptor session                 (PrdClose) */
/* - Open a print session and set a print descriptor    (PrtMgrOpen) */
/* - Set a valid and an invalid print option       (PrtMgrSetOption) */
/* - Get a list of all current print options     (PrtMgrListOptions) */
/* - Query each print option and print the value (PrtMgrQueryOption) */
/* - Start the print job                            (PrtMgrStartDoc) */
/* - Write buffers of data to the print job            (PrtMgrWrite) */
/* - End the print job.                              (PrtMgrEndDoc) */
/* - Close the print session                          (PrtMgrClose) */
/* - Destroy the created Prd                 (PrdDestroyDescriptor) */
/* - Terminate the PrintManager process           (PrtMgrTerminate) */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define  INCL_PRD
#define  INCL_PRTMGR
#define  INCL_ERRORS
#include <ekipmgr.h>

/*-------------------------------------------------------------------*/
/* Print descriptor name for MVS and VM, and OS/400.                  */
/* The PrdName for MVS and VM is used in this example.               */
/* To use the OS/400 PrdName, change PRDNAMEOS400 to PRDNAME.        */
/* Make sure you have access to write to the library you specify     */
/* in the group exact name below.                                    */
/*-------------------------------------------------------------------*/

#define  PRDNAME  "GRPEXACT=MYGROUP.PRD PRDNAME=SAMPLE PRD"
#define  PRDNAMEOS400  "GRPEXACT=QGPL/MYGROUP PRDNAME=SAMPLE PRD"
```

Figure B-20 (Part 1 of 13). Example of a PrintManager Print Session in C language

```
/*--------------------------------------------------------------------*/
/* Function declares                                                  */
/*--------------------------------------------------------------------*/

static void DisplayError(HAB, PSZ);

int main(void);


/*======================== Main Program Starts =======================*/

int main()
{
   /*-----------------------------------------------------------------*/
   /* General Declarations                                            */
   /*-----------------------------------------------------------------*/

   USHORT      i;                    /* index for loop              */
   BOOL        rc;                   /* return code                 */
   HAB         hab;                  /* anchor-block handle         */
   HPRD        hprd;                 /* handle for Prd edit session*/
   HPRM        hprm;                 /* handle for print session    */
   PSZ         prdname;              /* name of print descriptor    */
   OPTDEFN2    optdefn;              /* OPTDEFN2 structure          */
   OPTDATA1    setoptdata;           /* OPTDATA1 structure          */
   OPTDATA1    *optdata;             /* pointer to OPTDATA1 struct.*/
   PBYTE       Buffer;               /* pointer to data buffer      */
   LONG        buflen;               /* length of data in buffer    */
   PBYTE       listbuffer;           /* pointer to list opt buffer */
   PBYTE       qrybuffer;            /* pointer to query opt buffer*/
   LONG        length;               /* length of data in buffer    */
   ULONG       level;                /* level of data in buffer     */
   PSZ         *optnames;            /* pointer to opt names buffer*/
   ULONG       lengthNeeded;         /* length needed for buffer    */
   ULONG       itemsReturned;        /* # items returned in buffer */
   ULONG       itemsRemaining;       /* # items remaining in buffer*/
   ULONG       controlPrd;           /* Prd control parameter       */
   ULONG       controlGrp;           /* Prd group control parameter*/

   /*-----------------------------------------------------------------*/
   /* Declares for data written to the print job.                     */
   /*-----------------------------------------------------------------*/

   PBYTE     data1 ="First buffer of data written using PrtMgrWrite.";
   PBYTE     data2 ="Second buffer of data written using PrtMgrWrite.";
   LONG      datalen1 = strlen(data1);
   LONG      datalen2 = strlen(data2);
```

Figure B-20 (Part 2 of 13). Example of a PrintManager Print Session in C language

```
/*============================================================*/
/* Start a PrintManager process                (PrtMgrInitialize) */
/*============================================================*/

hab = PrtMgrInitialize(0);

if (hab == NULL)
   {
   printf("\n  ERROR issued by PrtMgrInitialize\n");
   return(0);
   }

/*============================================================*/
/* Open a print descriptor session                    (PrdOpen) */
/*============================================================*/

hprd = PrdOpen(hab);

if (hprd == NULL)
   DisplayError(hab, "PrdOpen");

/*============================================================*/
/* Set rules and valid values for options          (PrdSetOption) */
/*============================================================*/

/*------------------------------------------------------------*/
/* Define buffer level as using an OPTDEFN2 structure and define  */
/* constant values used in OPTDEFN2 structure for setting options.*/
/*------------------------------------------------------------*/

level = 2;                          /* using OPTDEFN2 structure   */
buflen = (LONG)sizeof(OPTDEFN2);    /* length of buffer passed in */
Buffer = (PBYTE)&optdefn;           /* pointer to OPTDEFN2 buffer */

optdefn.flDefType = PRTMGR_STRL;    /* default in string format   */
optdefn.cbDefLength = 0;            /* not used for string format */
```

Figure B-20 (Part 3 of 13). Example of a PrintManager Print Session in C language

```
/*-------------------------------------------------------------*/
/* Set DUPLEX ...                                              */
/*-------------------------------------------------------------*/

optdefn.pszName = "DUPLEX";          /* option name = DUPLEX       */
optdefn.pbDef = "NO";                /* default value = NO         */
optdefn.flRule = PRD_LIST;           /* rule type is LIST          */
optdefn.pszVv = "YES, NO, TUMBLE";   /* list of valid values is    */
                                     /*    YES, NO, TUMBLE         */

rc = PrdSetOption(hprd,
                  level,
                  buflen,
                  Buffer);

if (rc == FALSE)
   DisplayError(hab, "PrdSetOption");

/*-------------------------------------------------------------*/
/* Set COPIES option ...                                       */
/*-------------------------------------------------------------*/

optdefn.pszName = "COPIES";          /* option name = COPIES       */
optdefn.pbDef = "1";                 /* default value = 1          */
optdefn.flRule = PRD_RANGE;          /* rule type is RANGE         */
optdefn.pszVv = "0, 1, 255";         /* range of valid values is   */
                                     /*    precision = 0           */
                                     /*    minimum = 1             */
                                     /*    maximum = 255           */

rc = PrdSetOption(hprd,
                  level,
                  buflen,
                  Buffer);

if (rc == FALSE)
   DisplayError(hab, "PrdSetOption");
```

Figure B-20 (Part 4 of 13). Example of a PrintManager Print Session in C language

```
/*==================================================================*/
/* Save the print descriptor                    (PrdSaveDescriptor) */
/*==================================================================*/

level = 1;                          /* Buffer contains Prd name   */
Buffer = PRDNAME;                   /* set print descriptor name  */
length = strlen(PRDNAME);           /* size of Buffer             */
controlPrd = PRD_CREATE_OR_UPDATE;  /* Always update Prd          */
controlGrp = PRD_AUTO_CREATE;       /* Create new group           */

rc = PrdSaveDescriptor(hprd,
                       level,
                       length,
                       Buffer,
                       controlPrd,
                       controlGrp);

if (rc == FALSE)
   DisplayError(hab, "PrdSaveDescriptor");

/*==================================================================*/
/* Close the print descriptor session                  (PrdClose) */
/*==================================================================*/

rc = PrdClose(hprd);

if (rc == FALSE)
   DisplayError(hab, "PrdClose");


/*==================================================================*/
/* Open a print session and set a print descriptor   (PrtMgrOpen) */
/*==================================================================*/

hprm = PrtMgrOpen(hab,
                  PRDNAME,
                  0L,
                  NULL);

if (hprm == NULL)
   DisplayError(hab, "PrtMgrOpen");
```

Figure B-20 (Part 5 of 13). Example of a PrintManager Print Session in C language

```
/*=================================================================*/
/* Set a valid print option                      (PrtMgrSetOption) */
/*=================================================================*/

/*-----------------------------------------------------------------*/
/* Define buffer level as using an OPTDATA1 structure and define  */
/* constant values used in OPTDATA1 structure for setting options.*/
/*-----------------------------------------------------------------*/

level = 1;                          /* using OPTDATA1 structure   */
buflen = (LONG)sizeof(OPTDATA1);    /* length of buffer passed in */
Buffer = (PBYTE)&setoptdata;        /* pointer to OPTDATA1 buffer */

setoptdata.flValType = PRTMGR_STRL; /* value is in string format  */
setoptdata.cbValLength = 0;         /* not used for string format */

/*-----------------------------------------------------------------*/
/* Set DUPLEX option ...                                           */
/*-----------------------------------------------------------------*/

prdname = NULL;                     /* no new print descriptor    */
setoptdata.pszOption = "DUPLEX";    /* option name = DUPLEX       */
setoptdata.pbValue = "YES";         /* invalid value = MAYBE      */

rc = PrtMgrSetOption(hprm,
                     prdname,
                     level,
                     buflen,
                     Buffer);

if (rc == FALSE)
   DisplayError(hab, "PrtMgrSetOption");


/*=================================================================*/
/* Set an invalid print option                  (PrtMgrSetOption) */
/*=================================================================*/

printf ("\nSetting an invalid print option...\n");
printf ("An error message should be displayed...\n");
```

Figure B-20 (Part 6 of 13). Example of a PrintManager Print Session in C language

```
/*-------------------------------------------------------------*/
/* Set COPIES option to an invalid value ...                   */
/* Since the valid range in the print descriptor is 1 to 255, 300 */
/* is invalid and the value of COPIES will remain the default  */
/* value from the print descriptor which is 1.                 */
/*-------------------------------------------------------------*/

setoptdata.pszOption = "COPIES";    /* option name = COPIES    */
setoptdata.pbValue = "300";         /* value = 300             */

rc = PrtMgrSetOption(hprm,
                     NULL,
                     level,
                     buflen,
                     Buffer);

if (rc == FALSE)
   DisplayError(hab, "PrtMgrSetOption");


/*=============================================================*/
/* List Options                                (PrtMgrListOptions) */
/*=============================================================*/

/*-------------------------------------------------------------*/
/* Invoke first time to get the size of the listbuffer needed. */
/*-------------------------------------------------------------*/

level = 1;                          /* returns array of STRL type.*/
length = 0;                         /* set to zero and NULL to get*/
listbuffer = NULL;                  /* the actual buffer length.  */

rc = PrtMgrListOptions(hprm,
                       level,
                       length,
                       listbuffer,
                       &lengthNeeded,
                       &itemsReturned,
                       &itemsRemaining,
                       NULL);

if(rc == FALSE)
   DisplayError(hab, "PrtMgrListOptions");
```

Figure B-20 (Part 7 of 13). Example of a PrintManager Print Session in C language

```
/*-------------------------------------------------------------*/
/* Allocate space needed and invoke again to get option names.  */
/* If there is a memory allocation error, abort the session,    */
/* terminate PrintManager and return.                           */
/*-------------------------------------------------------------*/
if ((listbuffer = (PBYTE) malloc( (USHORT) lengthNeeded )) == NULL)
   {
   printf("\nERROR memory allocation failure");
   PrtMgrAbort(hprm);
   PrtMgrTerminate(hab);
   return(0);
   }

length = lengthNeeded;              /* length of the buffer      */

rc = PrtMgrListOptions(hprm,
                       level,
                       length,
                       listbuffer,
                       &lengthNeeded,
                       &itemsReturned,
                       &itemsRemaining,
                       NULL);

if(rc == FALSE)
   DisplayError(hab, "PrtMgrListOptions");

optnames = (PSZ *)listbuffer;       /* array of option names     */

/*=============================================================*/
/* Query Options                                (PrtMgrQueryOption) */
/*=============================================================*/

/*-------------------------------------------------------------*/
/* Invoke first time to get the maximum size of buffer needed   */
/* for the largest option in the set of current options.        */
/*-------------------------------------------------------------*/

level = 1;                          /* using OPTDATA1 structure.  */
length = 0;                         /* set to get max. length.    */
qrybuffer = NULL;

rc = PrtMgrQueryOption(hprm,
                       level,
                       length,
                       qrybuffer,
                       &lengthNeeded);

if(rc == FALSE)
   DisplayError(hab, "PrtMgrQueryOption");
```

Figure B-20 (Part 8 of 13). Example of a PrintManager Print Session in C language

```
/*-------------------------------------------------------------*/
/* Allocate storage for the largest print option.              */
/* If there is a memory allocation error, abort the session,  */
/* terminate PrintManager and return.                         */
/*-------------------------------------------------------------*/

if ((qrybuffer = (PBYTE) malloc( (USHORT) lengthNeeded ) ) == NULL)
   {
   printf("\nERROR memory allocation failure");
   PrtMgrAbort(hprm);
   PrtMgrTerminate(hab);
   return(0);
   }

length = lengthNeeded;              /* length of the qrybuffer    */
optdata = (OPTDATA1 *) qrybuffer;   /* points to OPTDATA1 type.   */
                                    /* query all options from list*/

/*-------------------------------------------------------------*/
/* Query each option for it's value.                          */
/*-------------------------------------------------------------*/

printf ("\nList of current print options:\n");

for (i = 0; i < (USHORT) itemsReturned; i++)
   {
   optdata->pszOption = optnames[i];/* set option name in OPDATA1*/

   rc = PrtMgrQueryOption(hprm,
                          level,
                          length,
                          qrybuffer,
                          &lengthNeeded);
   if(rc == FALSE)
      DisplayError(hab, "PrtMgrQueryOption");

   /*----------------------------------------------------------*/
   /* Print the options to the screen                          */
   /*----------------------------------------------------------*/
   if(optdata->pszOption != NULL)
      printf("\n %s", optdata->pszOption);

   if(optdata->pbValue != NULL)
      printf("\n      Value  =  %s\n", optdata->pbValue);
   }

free(listbuffer);
free(qrybuffer);
```

Figure B-20 (Part 9 of 13). Example of a PrintManager Print Session in C language

```
/*==================================================================*/
/* Start the print job                              (PrtMgrStartDoc) */
/*==================================================================*/

rc = PrtMgrStartDoc(hprm,
                    NULL);

if (rc == FALSE)
   DisplayError(hab, "PrtMgrStartDoc");

/*==================================================================*/
/* Write buffers of data to the print job            (PrtMgrWrite) */
/*==================================================================*/

rc = PrtMgrWrite(hprm,
                 datalen1,
                 data1);

if (rc == FALSE)
   DisplayError(hab, "PrtMgrWrite");

rc = PrtMgrWrite(hprm,
                 datalen2,
                 data2);

if (rc == FALSE)
   DisplayError(hab, "PrtMgrWrite");

/*==================================================================*/
/* End the print job                                (PrtMgrEndDoc) */
/*==================================================================*/

rc = PrtMgrEndDoc(hprm) != 0;

if (rc == FALSE)
   DisplayError(hab, "PrtMgrEndDoc");
```

Figure B-20 (Part 10 of 13). Example of a PrintManager Print Session in C language

```
   /*==================================================================*/
   /* Close the print session                          (PrtMgrClose) */
   /*==================================================================*/

   rc = PrtMgrClose(hprm);

   if (rc == FALSE)
      DisplayError(hab, "PrtMgrClose");

   /*==================================================================*/
   /* Destroy the created Prd                (PrdDestroyDescriptor) */
   /*==================================================================*/

   controlGrp = PRD_AUTO_DELETE;        /* delete empty Prd group    */

   rc = PrdDestroyDescriptor(hab,
                             PRDNAME,
                             controlGrp);

   if (rc == FALSE)
      DisplayError(hab, "PrdDestroyDescriptor");

   /*==================================================================*/
   /* Terminate the PrintManager process          (PrtMgrTerminate) */
   /*==================================================================*/

   rc = PrtMgrTerminate(hab);

   if (rc == FALSE)
      DisplayError(hab, "PrtMgrTerminate");

   return(0);
}
```

Figure B-20 (Part 11 of 13). Example of a PrintManager Print Session in C language

```
/*********************************************************************/
/*                    Display Error Function                         */
/*                                                                   */
/* The input to this function is the hab parameter value returned    */
/* from PrtMgrInitialize and a string indicating the last verb       */
/* issued. This function displays the name of the last verb issued   */
/* and the error code set by the verb. If additional error           */
/* information is available this information is also displayed.      */
/*********************************************************************/

static void DisplayError(HAB hab, PSZ verbname)
{
   BOOL         rc;
   ERRINFO     *errinfo;
   ERRMSG       errmsg;
   ULONG        total;
   LONG         index;

   printf("\n  ERROR issued by %s", verbname);

   /*=================================================================*/
   /* Display the error information             (PrtMgrGetErrorInfo) */
   /*                                          (PrtMgrGetExtErrorMsg) */
   /*=================================================================*/

   errinfo = PrtMgrGetErrorInfo(hab);
   if (errinfo)
      {
      /*-------------------------------------------------------------*/
      /* The idError field in the errinfo structure returned by      */
      /* PrtMgrGetErrorInfo contains the error code in the 2 low     */
      /* order bytes and the severity in the 2 high order bytes.     */
      /* LOUSHORT and HIUSHORT are macros defined in the             */
      /* PrintManager header files.                                  */
      /*-------------------------------------------------------------*/

      printf("\n\n  Error Code = %x:    Severity = %x\n",
             LOUSHORT(errinfo->idError),
             HIUSHORT(errinfo->idError));

      /*-------------------------------------------------------------*/
      /* Invoke PrtMgrGetExtErrorMsg with an index value of 0 to get */
      /* the total number of additional error messages.              */
      /*-------------------------------------------------------------*/

      rc = PrtMgrGetExtErrorMsg(errinfo,
                                0,
                                &errmsg,
                                &total);
```

Figure B-20 (Part 12 of 13). Example of a PrintManager Print Session in C language

```
      /*-------------------------------------------------------------*/
      /* Display the error identifier and error value for each       */
      /* additional error message.                                   */
      /*-------------------------------------------------------------*/

      for (index = 1; index <= total; index++)
         {
         rc = PrtMgrGetExtErrorMsg(errinfo,
                                   index,
                                   &errmsg,
                                   &total);
         if (rc)
            {
            printf("\n    Error identifier: %lx\n",
                   errmsg.idIdentifier);

            printf("    Error value:  %s\n", errmsg.szValue);
            }
         }

      /*-------------------------------------------------------------*/
      /* The storage for the ERRINFO structure returned by           */
      /* PrtMgrGetErrorInfo must be freed using PrtMgrFreeErrorInfo. */
      /*-------------------------------------------------------------*/

      PrtMgrFreeErrorInfo(errinfo);
      }

   printf("\n");
   return;
}
```

Figure B-20 (Part 13 of 13). Example of a PrintManager Print Session in C language

Appendix C. PrintManager COBOL Applications

This appendix provides the following information about PrintManager COBOL applications:

- COBOL language coding conventions
- Using PrintManager error verbs
- PrintManager Interface data types for the COBOL language
- PrintManager Interface COBOL language verb reference
- COBOL language example.

Note: PrintManager Interface COBOL language support is available in OS/400 only.

COBOL Language Coding Conventions

Copy Files

The copy file (EKICONC) contains definitions of the constants used to interface with PrintManager. The PrintManager/400 copy file for COBOL is shipped with the IBM COBOL/400 compiler. Refer to *AS/400 Languages: COBOL/400 User's Guide*, SC09-1158, for more information.

Interdependent Parameters

With some PrintManager Interface verbs, the size of one parameter is dependent on the value of another parameter. In this case, the size of the dependent parameter is represented by a name in lower case, where the name in lower case is the name of the parameter that controls the size. You must specify an appropriate literal value in this name and the size parameter cannot exceed the specified value.

The following example (on the SPROPEN call) shows how a string and a input length are represented:

```
* Prd Name input length
 77 PRDNAMEL    PIC S9(9) USAGE BINARY.
* Prd Name
 77 PRDNAME     PIC X(prdnamel).
```

In this example, the word "prdnamel" in lower case indicates that the current value of the variable PRDNAMEL must not exceed the declared length of the character string PRDNAME.

Length Parameters

For COBOL, there are three types of length parameters: input length, area length, and data length.

input length Identifies the length of data found in another *input* parameter. The length includes all significant data and may include trailing blanks (which are ignored). An example of an input length is the *PRDNAME* parameter passed to the SPROPEN verb.

area length Identifies the length of an *output* parameter. This value represents the number of bytes available to PrintManager to return data in this output parameter. An example of an area length is the *VALAL* parameter passed to the SPRQOPT verb.

data length Identifies the length of the significant data returned in a parameter. An example of a data length is the *VALDL* parameter passed to the SPRQOPT verb.

The following example shows how the input length, area length, and data length parameters are used. For example, if you want to query the value of the **DATATYPE** print option, which is currently set to **AFPDS**:

1. The option name variable defined in the program is 10 characters long. You store the string "DATATYPE" in this variable before calling SPRQOPT. The option name input length parameter can be set to 8 (the length of the string "DATATYPE"). It could also be set to 10 if the string "DATATYPE" is padded with blanks in the option name variable. You could set the input length parameter to the total size of the variable at the start of the program if you knew that the stored option name would always be padded with blanks.

2. The variable to receive the print option value is 30 characters long. Therefore, you store 30 in the print option value area length parameter because that is the maximum number of characters your program can receive.
3. The SPRQOPT verb is now invoked.
4. The print option value returned is "AFPDS". The print option value data length parameter is set to 5 by PrintManager because that is the length of the string that was returned to your program.

If an area is too small to hold a returned value on the SPRQOPT verb, you will receive error code 16554 (X'40AA') because the buffer is too small to hold the requested data. The data length parameter for that area is set to the total amount of data that would be returned if the area was large enough. In this case, you must change the size of the area and the value of the area length parameter and recompile the program.

Using PrintManager Error Verbs

Except for the SPRINIT and SPRTERM verbs, if a PrintManager verb returns a PMGR-FALSE or 0 return value, you can use the PrintManager error verbs to obtain information about the error as follows:

- Use the SPRGERI verb to get the error information associated with the last PrintManager Interface error. The error information returned includes the error code and the error severity.

  A severity code of 4 indicates a warning condition, while a severity code of 8 indicates an error condition. A warning condition means that the requested function completed successfully, and you may continue, although results may not be as expected. For example, an error of X'40B1' (16561) has a severity

code of 4, and indicates that truncation occurred when a valid value was set in a print descriptor. An error condition means that an error occurred, and the requested function did not complete successfully.

- For some errors, additional error information can be obtained by using the error information handle as input to SPRGEEM. SPRGEEM returns 3 output parameters containing the following:

  MSGID Provides additional information about the error.

  ERRDL The length of the error message returned in the ERRMSG parameter.

  ERRMSG Provides additional error information.

  Note: Appendix F, Verb Error Codes lists PrintManager error codes and provides general explanations of associated errors, a list of PrintManager Interface verbs that may have issued the error, user responses, and a list of additional messages (if any) that are issued.

- After the error information handle from SPRGERI is no longer needed, use the SPRFERI verb to free internal error information storage and clear the error information handle.

Data Types for COBOL Language

This section describes the data types for COBOL language, which are similar to the general data types described in "PrintManager Interface Verb Data Types" on page 4-2. Each description of a COBOL data type, therefore, provides data characteristics or mappings that are unique to the COBOL language.

Data Type	Description

BOOL	(See LONG)
	The constants PMGR-TRUE and PMGR-FALSE are defined in the EKICONC copy file as:
	`77 PMGR-TRUE VALUE IS 1.`
	`77 PMGR-FALSE VALUE IS 0.`

BUFFER	Buffer. In general, the BUFFER type maps to an input length and a variable area.
BYTE	(See CHAR)
CHAR	A single-byte character that maps to the COBOL "PIC X" type.
ERRINFO	Error information handle used with the SPRGERI verb to return error information.

ERRMSG	The SPRGEEM verb returns additional error information in the following output parameters:
	`* Error ID`
	`77 ERRID PIC S9(9) USAGE BINARY.`
	`* Error message data length`
	`77 ERRDL PIC S9(9) USAGE BINARY.`
	`* Error message`
	`77 ERRMSG PIC X(256).`

ERRORID	Consists of the following two parameters:

```
* Error Severity
 77 ERROR-SEV   PIC S9(9) USAGE BINARY.
* Error Code
 77 ERROR-CODE  PIC S9(9) USAGE BINARY.
```

HAB

A unique identifier (handle) for a PrintManager anchor block

HPRM

A print session identifier.

LONG

A 32-bit integer that maps to COBOL "PIC S9(9) USAGE BINARY".

OPTDATA1

This type maps to the following parameters on the SPRSOPT call:

```
* Option name input length
 77  OPTL       PIC S9(9) USAGE BINARY.
* Option name
 77  OPT        PIC X(optl).
* Value type
 77  VALTYP     PIC S9(9) USAGE BINARY.
* Print option value input length
 77  VALL       PIC S9(9) USAGE BINARY.
* Print option value
 77  VAL        PIC X(vall).
```

On the SPRQOPT call, OPTDATA1 maps to the following (note the additional VALDL parameter):

```
* Option name input length
 77  OPTL       PIC S9(9) USAGE BINARY.
* Option name
 77  OPT        PIC X(optl).
* Value type
 77  VALTYP     PIC S9(9) USAGE BINARY.
* Print option value area length
 77  VALAL      PIC S9(9) USAGE BINARY.
* Print option value data length
 77  VALDL      PIC S9(9) USAGE BINARY.
* Print option value
 77  VAL        PIC X(valal).
```

NOTE: If the print option value is type *PMGR-BINARY*, then the data in the value parameter cannot be directly processed in COBOL. Also, the *VALL* parameter on SPRSOPT must be set to the exact size of the value data since padded blanks may be invalid for a binary value.

SDF

A self-defining field that maps to the "continue flag" variable in COBOL:

```
* Continue flag
 77  CONTINUE   PIC S9(9) USAGE BINARY.
```

This field is set to *PMGR-GET-FIRST* before the first in one or more of a series of SPRLOPT calls and is otherwise not referenced by the COBOL application.

SHORT (See LONG)

STRL A string with an implicit length count.

This type maps to one of the following:

```
* Input parameter
* String input length
 77 STRGL           PIC S9(9) USAGE BINARY.
* String
 77 STRG            PIC X(strgl).


* Output parameter
* String area length
 77 STRGL           PIC S9(9) USAGE BINARY.
* String data length
 77 STRGDL          PIC S9(9) USAGE BINARY.
* String
 77 STRG            PIC X(strgl).
```

The first form is used when the string is passed as an input parameter, and the second form is used when the string is an output parameter.

ULONG (See LONG)

USHORT (See LONG)

SPRABRT (Abort)

MVS	VM	OS/400	OS/2
		X	

Function

Aborts the print session.

Syntax

```
 CALL "SPRABRT" USING  HPRM
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-1. COBOL Syntax of the SPRABRT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use the SPRABRT verb to abort the print session and to delete the print job (if active). When the SPRABRT verb is issued, the *hprm* from the aborted session is no longer valid.

Ending your application without calling SPRCLOS or SPRABRT could cause unpredictable results.

Return Values

SUCCESS
A nonzero value means the print session was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADOC (Abort Document)

MVS	VM	OS/400	OS/2
		X	

Function

Aborts and deletes the print job.

Syntax

```
 CALL "SPRADOC" USING  HPRM
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-2. COBOL Syntax of the SPRADOC Verb

Parameters

HPRM (HPRM)
(Input). Identifies a valid print session.

Usage

Use the SPRADOC verb to abort processing of the print job and to delete it. When the SPRADOC verb is issued, PrintManager returns to Session Active state.

SPRADOC does not abort a print job that has already been ended with the SPREDOC verb.

Return Values

***SUCCESS* (BOOL)**
A nonzero value means the print job was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADDF (Add File)

MVS	VM	OS/400	OS/2
		X	

Function

Writes a file of print data to the print job.

Syntax

```
 CALL "SPRADDF" USING  HPRM
                       FILNAMEL
                       FILNAME
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* File Name input length
 77  FILNAMEL   PIC S9(9) USAGE BINARY.
* File Name
 77  FILNAME    PIC X(filnamel).
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-3. COBOL Syntax of the SPRADDF Verb

Parameters

HPRM
(Input). Identifies a valid print session.

FILNAMEL
(Input). The length of the *FILNAME* parameter.

FILNAME
(Input). A string containing the name of a file with print data to be sent to the spool. The file name must follow the naming convention for OS/400.

Usage

Use the SPRADDF verb to write a file of print data to the print job.

If an error occurs, PrintManager returns to Session Active state with the exception of two errors: 16412 ("cannot open file") and 16547 ("read error"). These two errors will leave you in Job-in-Progress state. See "PrintManager Interface Operating States" on page 3-5 for more information on operating states.

Note: The file specified in the *FILNAME* parameter must be a physical data file with FILETYPE (*DATA) and LVLCHK (*NO).

Return Values

***SUCCESS* (BOOL)**

A nonzero value means the named file was written to the print job, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRCLOS (Close)

MVS	VM	OS/400	OS/2
		X	

Function

Ends the current print session.

Syntax

```
 CALL "SPRCLOS" USING  HPRM
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-4. COBOL Syntax of the SPRCLOS Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use the SPRCLOS verb to end a print session and to free storage associated with the session. When SPRCLOS is issued, the *hprm* parameter from the closed session is no longer valid.

Ending your application without calling SPRCLOS or SPRABRT could cause unpredictable results.

Return Values

SUCCESS
A nonzero value means the print session ended successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPREDOC (End Document)

MVS	VM	OS/400	OS/2
		X	

Function

Ends the print job.

Syntax

```
 CALL "SPREDOC" USING  HPRM
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-5. COBOL Syntax of the SPREDOC Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use SPREDOC to end a print job. When the SPREDOC verb is issued, PrintManager returns to Session Active state.

Return Values

SUCCESS
A nonzero value means the print job ended successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRFERI (Free Error Information)

MVS	VM	OS/400	OS/2
		X	

Function

Frees storage associated with the error-information handle returned from the SPRGERI verb.

Syntax

```
 CALL "SPRFERI" USING  ERRORINFO
                       SUCCESS.

* Error Information handle
 77  ERRORINFO  PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-6. COBOL Syntax of the SPRFERI Verb

Parameters

ERRORINFO
(Input). The **ERRORINFO** handle.

Usage

Use SPRFERI to free storage for the error-information handle you specify in **ERRORINFO**. You should issue SPRFERI for every error-information handle returned with the SPRGERI verb.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

SUCCESS
A nonzero value means storage was freed successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGEEM (Get Error Message)

MVS	VM	OS/400	OS/2
		X	

Function

Gets additional error information associated with the error-information handle.

Syntax

```
 CALL "SPRGEEM" USING  ERRORINFO
                       INDX
                       MSGID
                       ERRDL
                       ERRMSG
                       TOTCOUNTER
                       SUCCESS.

* Error Information handle
 77  ERRORINFO  PIC S9(9) USAGE BINARY.
* Index into error information
 77  INDX       PIC S9(9) USAGE BINARY.
* Message ID
 77  MSGID      PIC S9(9) USAGE BINARY.
* Error message data length
 77  ERRDL      PIC S9(9) USAGE BINARY.
* Error message
 77  ERRMSG     PIC X(256).
* Total number of error messages in error information
 77  TOTCOUNTER PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-7. COBOL Syntax of the SPRGEEM Verb

Parameters

ERRORINFO
(Input). The error-information handle (**ERRORINFO**) returned by SPRGERI.

INDX
(Input). Specifies which additional message to return. Use a value of 1 to select the first message, a value of 2 to select the second message, and so on.

MSGID
(Output). The message identifier.

ERRDL
(Output). The data length of the error message.

ERRMSG
(Output). The error message.

TOTCOUNTER
(Output). SPRGEEM updates this parameter to specify the number of additional messages associated with the error-information handle.

Usage

The SPRGEEM verb returns additional error information in the output parameters. The original error-information handle returned from SPRGERI must be passed to SPRGEEM.

Information returned from SPRGEEM is often useful in further resolving the cause of errors. For example, for error code 16412 ("cannot open file"), the *ERRMSG* parameter provides the name of the file where the error occurred. For operating system errors, PrintManager updates the *ERRMSG* parameter with error text provided by the operating system. Information in the *ERRMSG* parameter is truncated to 256 characters, if necessary.

There may be 1 to *n* additional error messages associated with the error information handle. Set *INDX* to 0 to get the total number of messages that can be returned. If the value returned in *TOTCOUNTER* is 0, there are no additional messages. If there are additional messages, set *INDX* to correspond to the message you want to return. If you want to return all additional messages, call SPRGEEM repeatedly, incrementing *INDX* until *TOTCOUNTER* is reached.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

SUCCESS
A nonzero value means the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGERI (Get Error Information)

MVS	VM	OS/400	OS/2
		X	

Function

Gets error information from the last error.

Syntax

```
 CALL "SPRGERI" USING  HAB
                       ERROR-SEV
                       ERROR-CODE
                       ERRORINFO.

* Application Anchor Block handle
 77  HAB        PIC S9(9) USAGE BINARY.
* Error Severity
 77  ERROR-SEV  PIC S9(9) USAGE BINARY.
* Error Code
 77  ERROR-CODE PIC S9(9) USAGE BINARY.
* Error Information handle
 77  ERRORINFO  PIC S9(9) USAGE BINARY.
```

Figure C-8. COBOL Syntax of the SPRGERI Verb

Parameters

HAB
(Input). The anchor-block handle returned from a successful call to SPRINIT. See Appendix E, "System-Specific Information" on page E-1 for more information on SPRINIT.

ERROR-SEV
(Output). Error severity.

ERROR-CODE
(Output). Error code.

ERRORINFO
(Output). The **ERRORINFO** handle.

Usage

Use the SPRGERI verb to get the error information associated with the last PrintManager Interface error. You can then use the error-information handle (**ERRORINFO**) as input to the SPRGEEM verb to get additional information (if any) about the error. To free storage for the error-information handle, use SPRFERI.

Notes:

1. If PrintManager cannot be initialized (with SPRINIT (Initialize PrintManager)), SPRGERI (Get Error Information) will not return an error-information buffer.
2. Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

ERRORINFO

The SPRGERI verb returns an error-information handle for the last error. This can be used with the SPRGEEM (Get Error Message) verb to obtain more information about the error. See Appendix F, "Verb Error Codes" on page F-1 for information on PrintManager Interface verb error codes. A return value of 0 indicates that no previous error occurred.

As well as errors from PrintManager Interface verbs, you can get errors from API verbs because some PrintManager Interface verbs invoke PrintManager API verbs.

Environment Restrictions

None.

SPRINIT (Initialize PrintManager)

MVS	VM	OS/400	OS/2
		X	

Function

Initialize PrintManager.

Syntax

```
 CALL "SPRINIT" USING  OPTIONS
                       HAB.

* Initialization options
 77  OPTIONS    PIC S9(9) USAGE BINARY.
* Application Anchor Block handle
 77  HAB        PIC S9(9) USAGE BINARY.
```

Figure C-9. COBOL Syntax of the SPRINIT Verb

Parameters

OPTIONS
(Input). This parameter is reserved and should have a value of 0.

Usage

Use the SPRINIT verb to initialize PrintManager. You cannot issue any other PrintManager verbs (including error verbs) until you have successfully initialized PrintManager.

Return Values

HAB
The SPRINIT verb should return an anchor-block handle (HAB). A return value of 0, however, indicates an error while initializing PrintManager.

Ensure that you save the HAB value that the SPRINIT verb returns. You will need to specify this value on other PrintManager verbs. You cannot issue two successive calls to SPRINIT. You must terminate PrintManager with the SPRTERM verb before reinitializing PrintManager.

Environment Restrictions

None.

SPRLOPT (List Options)

MVS	VM	OS/400	OS/2
		X	

Function

List current print options for a print session.

Syntax

```
CALL "SPRLOPT" USING  HPRM
                      COUNTER
                      OPTARRAY
                      ITEMRETN
                      ITEMREMN
                      CONTIN
                      SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Number of elements in OPTARRAY
 77  COUNTER    PIC S9(9) USAGE BINARY.
* Array of option names
 01  OPTARRAY.
   10  OPTARY   OCCURS counter.
*       Option Name
     15 OPTNAME    PIC X(31).
* Number of items returned
 77  ITEMRETN   PIC S9(9) USAGE BINARY.
* Number of items remaining
 77  ITEMREMN   PIC S9(9) USAGE BINARY.
* Continue flag
 77  CONTIN     PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-10. COBOL Syntax of the SPRLOPT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

COUNTER
(Input). Specifies the number of elements in *OPTARRAY*.

OPTARRAY
(Output). Returns a list of current print-option names.

ITEMRETN
(Output). Indicates the number of print-option names returned in *OPTARRAY*.

ITEMREMN
(Output). For a partial list, indicates the number of option names not yet listed; otherwise, *ITEMREMN* is 0.

CONTIN
(Input/Output). Used to get a list of options in a series of calls (see "Usage"). If you are not using this field for a series of calls, store **PMGR-GET-FIRST** in it.

Usage

Use the SPRLOPT verb to list the set of current options in a print session.

If you don't know the maximum number of current options that you will have in a print session, you can list all the options by using the *CONTIN* flag as follows:

- Before the first call to SPRLOPT, set *CONTIN* to **PMGR-GET-FIRST**. If all of the options will not fit in the option array, then *ITEMREMN* will be greater than zero, indicating that there are more names in the current options.
- Subsequent calls to SPRLOPT will result in more option names being returned in your option array. When *ITMREMN* is zero, all of the current options were returned in the option array.

  Note: Any changes to the print options with SPRLOPT during a series of calls using *CONTIN* will not be reflected in the partial lists returned.

Return Values

SUCCESS
A nonzero value means the list request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPROPEN (Open)

MVS	VM	OS/400	OS/2
		X	

Function

Opens and identifies a print session.

Syntax

```
 CALL "SPROPEN" USING  HAB
                       PRDNAMEL
                       PRDNAME
                       BUFFERL
                       BUFFER
                       HPRM.

* Application Anchor Block handle
 77  HAB         PIC S9(9) USAGE BINARY.
* Prd Name input length
 77  PRDNAMEL    PIC S9(9) USAGE BINARY.
* Prd Name
 77  PRDNAME     PIC X(prdnamel).
* Buffer input length (must be 0; BUFFER reserved for future use)
 77  BUFFERL     PIC S9(9) USAGE BINARY.
* Buffer (reserved for future use)
 77  BUFFER      PIC X.
* Handle for Print session
 77  HPRM        PIC S9(9) USAGE BINARY.
```

Figure C-11. COBOL Syntax of the SPROPEN Verb

Parameters

HAB
(Input). The anchor-block handle returned from a successful call to SPRINIT.

PRDNAMEL
(Input). The input length for the *PRDNAME* parameter.

PRDNAME
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored and no validation is done if the value is all blanks, a single asterisk (*), or *PRDNAMEL* is zero,

BUFFERL
(Input). This field is reserved and must have a value of zero.

BUFFER
(Input). This parameter is reserved.

Usage

Use the SPROPEN verb to start a print session. You can also use SPROPEN to define the set of current options by specifying a print descriptor.

Return Values

HPRM

The SPROPEN verb should return the session identifier (*hprm*). A return value of 0, however, means an error occurred. Ensure that you save the *hprm* value that the SPROPEN verb returns. You will need to specify this value on other PrintManager Interface verbs used in a session.

Environment Restrictions

None.

SPRQOPT (Query Option)

MVS	VM	OS/400	OS/2
		X	

Function

Query the current value of a print option.

Syntax

```
CALL "SPRQOPT" USING   HPRM
                       OPTL
                       OPT
                       VALTYP
                       VALAL
                       VALDL
                       VAL
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Option name input length
 77  OPTL       PIC S9(9) USAGE BINARY.
* Option name
 77  OPT        PIC X(optl).
* Value type
 77  VALTYP     PIC S9(9) USAGE BINARY.
* Print option value area length
 77  VALAL      PIC S9(9) USAGE BINARY.
* Print option value data length
 77  VALDL      PIC S9(9) USAGE BINARY.
* Print option value
 77  VAL        PIC X(valal).
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-12. COBOL Syntax of the SPRQOPT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

OPTL
(Input). Specify the print-option name input length, which can be up to 31 characters, in the *OPT* parameter. Any additional characters will be truncated.

OPT
(Input). Specify the print-option name.

VALTYP
(Output). The value type. If the value of *VALTYP* is **PMGR-STRL**, the data is a character string. If the value of *VALTYP* is **PMGR-BINARY**, the data is binary data with a length of *VALDL*.

VALAL
(Input). The print-option value area length.

VALDL
(Output). The length of data in the *VAL* parameter.

VAL
(Output). The print-option value.

Usage

Use the SPRQOPT verb to list the current values for a print option.

If it is determined from the *VALAL* parameter that *VAL* is not large enough to hold the current values for the specified print option, then the verb will fail. *VALDL* will be set to the total length of the actual option values, and as many values as possible is returned in *VAL*. The error code will be set to X'40AA' ("buffer too small"). Therefore, *VALDL* will be greater than the value you specified in *VALAL*. To correct this, make *VAL* larger and set *VALAL* to this larger value.

See "Length Parameters" on page C-1 for a general description of how length parameters are used.

Return Values

SUCCESS
A nonzero value means the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRSDOC (Start Document)

MVS	VM	OS/400	OS/2
		X	

Function

Starts a print job and specifies a document name for the print job.

Syntax

```
 CALL "SPRSDOC" USING  HPRM
                       DOCNAMEL
                       DOCNAME
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Document Name input length
 77  DOCNAMEL   PIC S9(9) USAGE BINARY.
* Document Name
 77  DOCNAME    PIC X(docnamel).
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-13. COBOL Syntax of the SPRSDOC Verb

Parameters

HPRM (HPRM)
(Input). Identifies a valid print session.

DOCNAMEL
(Input). Specify the *DOCNAME* input length.

DOCNAME
(Input). A string containing the document name for the print job. If no document name is specified, the name is inherited from the previous print job in a session. If no previous document name exists, the operating system provides a default. The *DOCNAME* parameter must follow OS/400 naming conventions, and is truncated to 10 characters.

Usage

Use SPRSDOC to start a print job. You must issue SPREDOC before issuing another SPRSDOC within the same print session.

If the SPRSDOC verb is successful, PrintManager is placed in Job-in-Progress state.

Return Values

SUCCESS

A nonzero value means a new print job was initiated, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRSOPT (Set Option)

MVS	VM	OS/400	OS/2
		X	

Function

Set or change a print option.

Syntax

```
CALL "SPRSOPT" USING   HPRM
                       PRDNAMEL
                       PRDNAME
                       OPTL
                       OPT
                       VALTYP
                       VALL
                       VAL
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Prd Name input length
 77  PRDNAMEL   PIC S9(9) USAGE BINARY.
* Prd Name
 77  PRDNAME    PIC X(prdnamel).
* Option name input length
 77  OPTL       PIC S9(9) USAGE BINARY.
* Option name
 77  OPT        PIC X(optl).
* Value type
 77  VALTYP     PIC S9(9) USAGE BINARY.
* Print option value input length
 77  VALL       PIC S9(9) USAGE BINARY.
* Print option value
 77  VAL        PIC X(vall).
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-14. COBOL Syntax of the SPRSOPT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

PRDNAMEL
(Input). The input length of the *PRDNAME* parameter.

PRDNAME
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored if it is all blanks, or a single asterisk (*), or the *PRDNAME* parameter is zero. If *DEL is specified, all print options are deleted from the set of current options, the print descriptor is no

longer in effect, and any options specified in the other parameters are not validated.

OPTL
(Input). Specify the print-option name input length, which can be up to 31 characters, in the *OPT* parameter. Any additional characters will be truncated.

OPT
(Input) Specify the print-option name.

VALTYP
(Input). The value type. If the value of *VALTYP* of **PMGR-STRL**, the data is a character string. If the value of *VALTYP* is **PMGR-BINARY**, the data is binary data with a length of *VALL*.

VALL
(Input). The length of data in the *VAL* parameter.

VAL
(Input). The print-option value.

Usage

Use the SPRSOPT verb to validate and set print-option information and to specify a print descriptor (see "Print Option Validation" on page 5-5).

If an error occurs in retrieving a print descriptor, any defaults and validation information from the previous print descriptor are no longer in effect.

If the *PRDNAME* parameter is specified as all blanks, an asterisk, or the *PRDNAMEL* parameter is zero, the options specified on this call are validated against the current print descriptor, if one has been previously defined. If no print descriptor has been defined, no validation is performed.

If a print descriptor is specified after print options have been set, the previously set print options are validated against the new print descriptor.

You should specify a print descriptor only once because each time you specify a print descriptor, it is reapplied and the set of current options is validated.

Return Values

SUCCESS
A nonzero value means the print-option values were set successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRTERM (Terminate PrintManager)

MVS	VM	OS/400	OS/2
		X	

Function

Terminate PrintManager.

Syntax

```
 CALL "SPRTERM" USING   HAB
                        SUCCESS.

* Application Anchor Block handle
 77  HAB         PIC S9(9) USAGE BINARY.
* Success/failure return code
 77  SUCCESS     PIC S9(9) USAGE BINARY.
```

Figure C-15. COBOL Syntax of the SPRTERM Verb

Parameters

HAB
(Input). The anchor-block handle returned from a successful call to SPRINIT.

Usage

Use the SPRTERM verb to terminate PrintManager and to release all associated resources. For the *hab* parameter, ensure that you enter the anchor-block handle returned on the SPRINIT verb when you initialized PrintManager. You must terminate a PrintManager session before you can reinitialize PrintManager.

Return Values

SUCCESS
A nonzero value means PrintManager was terminated successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRWRIT (Write)

MVS	VM	OS/400	OS/2
		X	

Function

Sends a buffer of data to the print job.

Syntax

```
 CALL "SPRWRIT" USING  HPRM
                       BUFL
                       BUFFER
                       SUCCESS.

* Handle for Print session
 77  HPRM       PIC S9(9) USAGE BINARY.
* Buffer input length (length of data to be written)
 77  BUFL       PIC S9(9) USAGE BINARY.
* Data to be written
* (BUFFER size (bufsize) must be greater than or equal to BUFL)
 77  BUFFER     PIC X(bufsize).
* Success/failure return code
 77  SUCCESS    PIC S9(9) USAGE BINARY.
```

Figure C-16. COBOL Syntax of the SPRWRIT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

BUFL
(Input). The length of the buffer of print data sent to the print job.

BUFFER
(Input). The buffer of print data sent to the print job.

Usage

Use the SPRWRIT verb to send a buffer of data to a print job. If an error occurs, PrintManager returns to Session Active state.

Return Values

SUCCESS

A nonzero value means the buffer of data was sent to the print job successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

Example of a PrintManager Session in COBOL Language

```
 IDENTIFICATION DIVISION.
*****************************************************************
*      PRINTMANAGER INTERFACE EXAMPLE --- COBOL                 *
*                                                               *
* This example shows how to use the PrintManager Print          *
* Interface. It is not intended to satisfy any particular       *
* functional needs.                                             *
*                                                               *
* Sequence of Operations:                                       *
*                                                               *
* - Start a PrintManager process                      (SPRINIT) *
* - Open a print session and set a print descriptor    (SPROPEN) *
* - Set a valid and an invalid print option            (SPRSOPT) *
* - Get a list of all current print options            (SPRLOPT) *
* - Query each print option and print the value        (SPRQOPT) *
* - Start the print job                               (SPRSDOC) *
* - Write buffers of data to the print job             (SPRADDF) *
* - End the print job.                                (SPREDOC) *
* - Close the print session                           (SPRCLOS) *
* - Terminate the PrintManager process                (SPRTERM) *
*                                                               *
*****************************************************************
```

Figure C-17 (Part 1 of 13). Example of a PrintManager Print Session in COBOL Language

```
 PROGRAM-ID. COBCPIEX.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM.
 OBJECT-COMPUTER. IBM.

*****************************************************************
*                                                               *
*                   DATA DIVISION                               *
*                                                               *
*****************************************************************
 DATA DIVISION.
 WORKING-STORAGE SECTION.

*---------------------------------------------------------------*
*  PrintManager Constants                                       *
*---------------------------------------------------------------*
      COPY EKICONC.

*---------------------------------------------------------------*
* Print descriptor name.                                        *
*                                                               *
* In this example note that the "universal" print descriptor    *
* name format is used.  This is to make the code as system-     *
* independent (and portable) as possible.                       *
*                                                               *
* This example also assumes that this print descriptor already  *
* exists and that it contains the following print options:      *
*                                                               *
*      Option Name: DUPLEX                                      *
*          Default:      YES                                    *
*          Rule:         List                                   *
*          Valid Values: YES, NO, TUMBLE                        *
*                                                               *
*      Option Name: COPIES                                      *
*          Default:      1                                      *
*          Rule:         Range                                  *
*          Valid Values: 1-255, zero decimal places             *
*---------------------------------------------------------------*
 77 CPRDNAME PIC X(33) VALUE IS
           "PRDNAME=IBM Print Manager Options".
```

Figure C-17 (Part 2 of 13). Example of a PrintManager Print Session in COBOL Language

```
*-----------------------------------------------------------------*
*  General Declarations                                           *
*-----------------------------------------------------------------*

* Error index and loop index
 77 I    PIC S9(9) USAGE BINARY.
*77 J    PIC S9(9) USAGE BINARY.

* Success/failure return code
 77 SUCCESS  PIC S9(9) USAGE BINARY.

* Current function name
 77 CURRFUN  PIC X(8).

* Anchor block handle
 77 HAB      PIC S9(9) USAGE BINARY.
 77 OPTIONS  PIC S9(9) USAGE BINARY.

* Handle for Print Session
 77 HPRM     PIC S9(9) USAGE BINARY.

* Print Descriptor input length and name
 77 PRDNAMEL PIC S9(9) USAGE BINARY VALUE 80.
 77 PRDNAME  PIC X(80).

* Reserved parameter for SPROPEN
 77 RSVDL    PIC S9(9) USAGE BINARY VALUE 0.
 77 RSVD     PIC X.

* Variables for starting a document
 77 DOCNAMEL PIC S9(9) USAGE BINARY VALUE 10.
 77 DOCNAME  PIC X(10).

* Variables for ending a document
 77 JOBID    PIC S9(9) USAGE BINARY.

* Counter of parameter structure for SPRLOPT
 77 COUNTER  PIC S9(9) USAGE BINARY.

* Number of items returned
 77 ITEMRETN PIC S9(9) USAGE BINARY.

* Number of items remaining
 77 ITEMREMN PIC S9(9) USAGE BINARY.
```

Figure C-17 (Part 3 of 13). Example of a PrintManager Print Session in COBOL Language

```
      * Option Data variables
       77 OPTL     PIC S9(9) USAGE BINARY VALUE 32.
       77 OPT      PIC X(32).
       77 VALTYP   PIC S9(9) USAGE BINARY.
       77 VALL     PIC S9(9) USAGE BINARY VALUE 80.
       77 VALDL    PIC S9(9) USAGE BINARY.
       77 VAL      PIC X(80).

      * Option Names array
       01 OPTNAMES.
         10 OPTARRAY OCCURS 50 INDEXED BY J.
      *       Option name
           15 OPTNAME   PIC X(31).

      * Continue variable
       77 CONTIN   PIC S9(9) USAGE BINARY.

      * Error verb parameter declares
       77 ERRINFO  PIC S9(9) USAGE BINARY.

      * Error message variables
       77 ERRID    PIC S9(9) USAGE BINARY.
       77 ERRDL    PIC S9(9) USAGE BINARY.
       77 ERRMSG   PIC X(256).

      * Error severity and error code
       77 ERRSEV  PIC S9(9) USAGE BINARY.
       77 ERRCOD  PIC S9(9) USAGE BINARY.

      * Total number of error messages in error information
       77 TOTCOUNTER PIC S9(9) USAGE BINARY.

      *-------------------------------------------------------------*
      *  Declares for data written to the print job.                *
      *-------------------------------------------------------------*
       77 DATALEN1 PIC S9(9) USAGE BINARY VALUE 42.
       77 DATA1    PIC X(43) VALUE IS
                   "First buffer of data written using SPRWRIT".
       77 DATALEN2 PIC S9(9) USAGE BINARY VALUE 43.
       77 DATA2    PIC X(44) VALUE IS
                   "Second buffer of data written using SPRWRIT".
```

Figure C-17 (Part 4 of 13). Example of a PrintManager Print Session in COBOL Language

```
*****************************************************************
*                                                               *
*                PROCEDURE DIVISION                             *
*                                                               *
*****************************************************************
 PROCEDURE DIVISION.

 MAINLINE.

*===============================================================*
* Start a PrintManager session                      (SPRINIT) *
*===============================================================*
     MOVE 0 TO OPTIONS.
     MOVE 0 TO HAB.
     CALL "SPRINIT" USING OPTIONS
                          HAB.
     MOVE HAB TO SUCCESS.
     MOVE "SPRINIT" TO CURRFUN.
     PERFORM CHKSUCC.

*===============================================================*
* Open a print session and set a print descriptor   (SPROPEN) *
*===============================================================*

     MOVE CPRDNAME TO PRDNAME.
     CALL "SPROPEN" USING HAB
                          PRDNAMEL
                          PRDNAME
                          RSVDL
                          RSVD
                          HPRM.
     MOVE HPRM TO SUCCESS.
     MOVE "SPROPEN" TO CURRFUN.
     PERFORM CHKSUCC.
```

Figure C-17 (Part 5 of 13). Example of a PrintManager Print Session in COBOL Language

```
*================================================================*
* Set a valid print option                              (SPRSOPT) *
*================================================================*

     PERFORM INIT-OPTDATA.

*----------------------------------------------------------------*
* Set DUPLEX option ...                                          *
*----------------------------------------------------------------*

* Use current print descriptor
     MOVE "*" TO PRDNAME.

     MOVE "DUPLEX" TO OPT.
     MOVE "YES" TO VAL.

     CALL "SPRSOPT" USING HPRM
                          PRDNAMEL
                          PRDNAME
                          OPTL
                          OPT
                          VALTYP
                          VALL
                          VAL
                          SUCCESS.

     MOVE "SPRSOPT" TO CURRFUN.
     PERFORM CHKSUCC.
```

Figure C-17 (Part 6 of 13). Example of a PrintManager Print Session in COBOL Language

```
*=============================================================*
* Set a invalid print option                        (SPRSOPT) *
*=============================================================*

     DISPLAY "Setting an invalid print option..." UPON SYSOUT.
     DISPLAY "An error message should be DISPLAYed." UPON SYSOUT.

*-------------------------------------------------------------*
* Set COPIES option to an invalid value ...                   *
* Since the valid range in the print descriptor is 1 to 255,  *
* 300 is invalid and the value of COPIES will remain the      *
* default value from the print descriptor which is 1.         *
*-------------------------------------------------------------*

     MOVE "COPIES" TO OPT.
     MOVE "300" TO VAL.

     CALL "SPRSOPT" USING HPRM
                          PRDNAMEL
                          PRDNAME
                          OPTL
                          OPT
                          VALTYP
                          VALL
                          VAL
                          SUCCESS.

     PERFORM CHKSUCC.
```

Figure C-17 (Part 7 of 13). Example of a PrintManager Print Session in COBOL Language

```
*=============================================================*
* List Options and Query Options          (SPRLOPT) (SPRQOPT) *
*=============================================================*

*-------------------------------------------------------------*
* LIST-OPTIONS invokes SPRLOPT to get a list of all options set *
* 50 at a time.  If there are more than 50 options, it is called*
* more than once.  It also calls QUERY-OPTION for each option   *
* listed.                                                       *
*-------------------------------------------------------------*

     PERFORM INIT-OPTARRAY VARYING J FROM 1 BY 1 UNTIL J = 50.
     MOVE 50 TO COUNTER.
     MOVE PMGR-GET-FIRST TO CONTIN.

     PERFORM LIST-OPTIONS TEST AFTER UNTIL ITEMREMN = 0.


*=============================================================*
* Start the print job                                (SPRSDOC) *
*=============================================================*

     MOVE 8 TO DOCNAMEL.
     MOVE "COBOLDOC" TO DOCNAME.
     CALL "SPRSDOC" USING HPRM
                          DOCNAMEL
                          DOCNAME
                          SUCCESS.

     MOVE "SPRSDOC" TO CURRFUN.
     PERFORM CHKSUCC.

*=============================================================*
* Write buffers of data to the print job             (SPRWRIT) *
*=============================================================*

     CALL "SPRWRIT" USING HPRM
                          DATALEN1
                          DATA1
                          SUCCESS.

     PERFORM CHKSUCC.

     CALL "SPRWRIT" USING HPRM
                          DATALEN2
                          DATA2
                          SUCCESS.

     MOVE "SPRWRIT" TO CURRFUN.
     PERFORM CHKSUCC.
```

Figure C-17 (Part 8 of 13). Example of a PrintManager Print Session in COBOL Language

```
*================================================================*
* End the print job                                    (SPREDOC) *
*================================================================*

     CALL "SPREDOC" USING HPRM
                          SUCCESS.

     MOVE "SPREDOC" TO CURRFUN.
     PERFORM CHKSUCC.

*================================================================*
* Close the print session                              (SPRCLOS) *
*================================================================*

     CALL "SPRCLOS" USING HPRM
                          SUCCESS.

     MOVE "SPRCLOS" TO CURRFUN.
     PERFORM CHKSUCC.

*================================================================*
* Terminate the PrintManager process                   (SPRTERM) *
*================================================================*

     CALL "SPRTERM" USING HAB
                          SUCCESS.

     MOVE "SPRTERM" TO CURRFUN.
     PERFORM CHKSUCC.
     STOP RUN.

 LIST-OPTIONS.
     CALL "SPRLOPT" USING HPRM
                          COUNTER
                          OPTNAMES
                          ITEMRETN
                          ITEMREMN
                          CONTIN
                          SUCCESS.

     MOVE "SPRLOPT" TO CURRFUN.
     PERFORM CHKSUCC.
```

Figure C-17 (Part 9 of 13). Example of a PrintManager Print Session in COBOL Language

```
*-------------------------------------------------------------*
* Query each option listed by SPRLOPT                         *
*-------------------------------------------------------------*

     DISPLAY "Listing current print options..." UPON SYSOUT.
     PERFORM INIT-OPTDATA.
     PERFORM QUERY-OPTIONS TEST AFTER VARYING J FROM 1 BY 1
                           UNTIL J = ITEMRETN.



 QUERY-OPTIONS.
*-------------------------------------------------------------*
*               Query and display options                     *
* Query each option returned in the optarray from SPRLOPT.    *
* Print each option and its value as it is returned from SPRQOPT*
*-------------------------------------------------------------*
     MOVE OPTNAME(J) TO OPT.
     CALL "SPRQOPT" USING HPRM
                          OPTL
                          OPT
                          VALTYP
                          VALL
                          VALDL
                          VAL
                          SUCCESS.

     MOVE "SPRQOPT" TO CURRFUN.
     PERFORM CHKSUCC.


*-------------------------------------------------------------*
* Print the options to the screen                             *
*-------------------------------------------------------------*
     DISPLAY "Option: " OPTNAME(J) UPON SYSOUT.
     DISPLAY "Value: " VAL UPON SYSOUT.

 INIT-OPTARRAY.
```

Figure C-17 (Part 10 of 13). Example of a PrintManager Print Session in COBOL Language

```
*-------------------------------------------------------------*
*              Initialize optarray for SPRLOPT                *
*                                                             *
* This function initializes the optarray optnamedl(s) TO      *
* zero and the optname(s) TO blanks.                          *
*-------------------------------------------------------------*

     INITIALIZE OPTNAME(J).

 INIT-OPTDATA.
*-------------------------------------------------------------*
* Define buffer level as using an OPTDATA1 structure and define *
* constant values used in OPTDATA1 structure for setting      *
* options.                                                    *
*-------------------------------------------------------------*

     MOVE PMGR-STRL TO VALTYP.
     INITIALIZE OPT.
     INITIALIZE VAL.

 CHKSUCC.
*-------------------------------------------------------------*
*                   Display Error Function                    *
*                                                             *
* This function uses the HAB parameter value returned from    *
* SPRINIT and a string indicating the last verb issued, CURRFUN.*
* The name of the last verb issued and the error code set BY  *
* the verb are displayed. If additional error information     *
* is available this information is also displayed.            *
*-------------------------------------------------------------*
```

Figure C-17 (Part 11 of 13). Example of a PrintManager Print Session in COBOL Language

```
*================================================================*
* Display the error information                        (SPRGERI) *
*                                                      (SPRGEEM) *
*================================================================*

     IF (SUCCESS = PMGR-FALSE)
        DISPLAY CURRFUN "  TEST FAILED!!    " UPON SYSOUT

* Get the error information handle
        MOVE 0 TO ERRINFO
        CALL "SPRGERI" USING HAB
                             ERRSEV
                             ERRCOD
                             ERRINFO
        IF (ERRINFO = 0)
          DISPLAY "  UNABLE TO GET ERROR INFO" UPON SYSOUT
          GOBACK
        END-IF
        DISPLAY "Error Code: " ERRCOD  UPON SYSOUT
        DISPLAY "Error Severity: " ERRSEV  UPON SYSOUT

*----------------------------------------------------------------*
* Invoke SPRGEEM with an index value of 0 to get the total       *
* number of additional error messages.                           *
*----------------------------------------------------------------*

        MOVE 0 TO I
        CALL "SPRGEEM" USING ERRINFO
                             I
                             ERRID
                             ERRDL
                             ERRMSG
                             TOTCOUNTER
                             SUCCESS
        MOVE 1 TO I
        PERFORM DISPLAY-ERROR-MESSAGE TOTCOUNTER TIMES
```

Figure C-17 (Part 12 of 13). Example of a PrintManager Print Session in COBOL Language

```
      *---------------------------------------------------------------*
      * The storage for the ERRINFO structure returned by             *
      * SPRGERI must be freed using SPRFERI.                          *
      *---------------------------------------------------------------*

             CALL "SPRFERI" USING ERRINFO
                                  SUCCESS

          END-IF.

       DISPLAY-ERROR-MESSAGE.
      *---------------------------------------------------------------*
      * Display the error identifier and error value for each         *
      * additional error message.                                     *
      *---------------------------------------------------------------*

          CALL "SPRGEEM" USING ERRINFO
                               I
                               ERRID
                               ERRDL
                               ERRMSG
                               TOTCOUNTER
                               SUCCESS.
          DISPLAY "   Error identifier: " ERRID UPON SYSOUT.
          DISPLAY "   Error value:  " ERRMSG UPON SYSOUT.
          DISPLAY "                 "  UPON SYSOUT.
          DISPLAY "                 "  UPON SYSOUT.
          ADD 1 TO I.
```

Figure C-17 (Part 13 of 13). Example of a PrintManager Print Session in COBOL Language

Appendix D. PrintManager RPG Applications

This appendix provides the following information about PrintManager RPG applications:

- RPG language coding conventions
- Using PrintManager error verbs
- PrintManager Interface data types for the RPG language
- PrintManager Interface RPG language verb reference
- RPG language example.

Note: PrintManager Interface RPG language support is available in OS/400 only.

RPG Language Coding Conventions

Copy Files

The copy file (EKICONR) contains definitions of the constants used to interface with PrintManager. The PrintManager/400 copy file for RPG is shipped with the IBM RPG/400 compiler. Refer to *AS/400 Languages: RPG/400 User's Guide*, SC09-1161, for more information.

Interdependent Parameters

With some PrintManager Interface verbs, the size of one parameter is dependent on the value of another parameter. In this case, the size of the dependent parameter is represented by "<x>," where *x* is the value in the size parameter. You must specify an explicit size in the dependent parameter, and the value of the size parameter cannot exceed the specified value.

The following example (on the SPROPEN call) shows how a string and a input length are represented:

```
I* Prd Name input length
I                                     B   1   40PRDNL
I* Prd Name
INAMVAR      DS
I                                         1 <x> PRDNAM
```

In this example, the string "<x>" indicates that the current value of the variable PRDNL must not exceed the declared length of the character string PRDNAM.

Length Parameters

For RPG, there are three types of length parameters: input length, area length, and data length.

input length Identifies the length of data found in another *input* parameter. The length includes all significant data and may include trailing blanks (which are ignored). In RPG, the input length should be set to the number of significant characters stored in the parameter to prevent problems in cases where blank padding has not been done. An example of an input length is the *PRDNL* parameter passed to the SPROPEN verb.

area length Identifies the length of an *output* parameter. This value represents the number of bytes available to PrintManager to return data in this output parameter. An example of an area length is the *VALL* parameter passed to the SPRQOPT verb.

data length Identifies the length of the significant data returned in a parameter. An example of a data length is the *VALDL* parameter passed to the SPRQOPT verb.

The following example shows how the input length, area length, and data length parameters are used. For example, if you want to query the value of the **DATATYPE** print option, which is currently set to **AFPDS**:

1. The option name variable defined in the program is 10 characters long. You store the string "DATATYPE" in this variable before calling SPRQOPT. The option name input length parameter can be set to 8 (the length of the string "DATATYPE"). It could also be set to 10 if the string "DATATYPE" is padded with blanks in the option name variable. You could set the input length parameter to the total size of the variable at the start of the program if you knew that the stored option name would always be padded with blanks.

2. The variable to receive the print option value is 30 characters long. Therefore, you store 30 in the print option value area length parameter because that is the maximum number of characters your program can receive.
3. The SPRQOPT verb is now invoked.
4. The print option value returned is "AFPDS". The print option value data length parameter is set to 5 by PrintManager because that is the length of the string that was returned to your program.

If an area is too small to hold a returned value on the SPRQOPT verb, you will receive error code 16554 (X'40AA') because the buffer is too small to hold the requested data. The data length parameter for that area is set to the total amount of data that would be returned if the area was large enough. In this case, you must change the size of the area and the value of the area length parameter and recompile the program.

Using PrintManager Error Verbs

Except for the SPRINIT and SPRTERM verbs, if a PrintManager verb returns a PFALSE or 0 return value, you can use the PrintManager error verbs to obtain information about the error as follows:

- Use the SPRGERI verb to get the error information associated with the last PrintManager Interface error. The error information returned includes the error code and the severity of the error.

  A severity code of 4 indicates a warning condition, while a severity code of 8 indicates an error condition. A warning condition means that the requested function completed successfully, and you may continue, although results may not be as expected. For example, an error of X'40B1' (16561) has a severity

code of 4, and indicates that truncation occurred when a valid value was set in a print descriptor. An error condition means that an error occurred, and the requested function did not complete successfully.

- For some errors, additional error information can be obtained by using the error information handle as input to SPRGEEM. SPRGEEM returns 3 output parameters containing the following:

  MSGID Provides additional information about the error.

  ERRDL The length of the error message returned in the ERRMSG parameter.

  ERRMSG Provides additional error information.

  Note: Appendix F, Verb Error Codes lists PrintManager error codes and provides general explanations of associated errors, a list of PrintManager Interface verbs that may have issued the error, user responses, and a list of additional messages (if any) that are issued.

- After the error information handle from SPRGERI is no longer needed, use the SPRFERI verb to free internal error information storage and clear the error information handle.

Data Types for RPG Language

This section describes the data types for RPG language, which are similar to the general data types described in "PrintManager Interface Verb Data Types" on page 4-2. Each description of a RPG data type, therefore, provides data characteristics or mappings that are unique to the RPG language.

Data Type **Description**

BOOL (See LONG)

The named constants *PTRUE* and *PFALSE* are defined in the EKICONR copy file as:

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7...
I* Constants
I              1                     C         PTRUE
I              0                     C         PFALSE
```

BUFFER Buffer

In general, the BUFFER type maps to an input length and a variable area.

BYTE (See CHAR)

CHAR A single-byte character that maps to the following RPG structure:

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7...
I* Char
IICHAR       DS
I                                          1   1 CHRVAR
```

ERRINFO Error information handle used with the SPRGERI verb to return error information.

ERRMSG The SPRGEEM verb returns additional error information in the following output parameters:

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7.
I* Message ID
IIDVAR      DS
I                                     B   1   40MSGID
I* Error message data length
IMLVAR      DS
I                                     B   1   40ERRDL
I* Error message
IERRMSG     DS
```

ERRORID Consists of the following two parameters:

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7.
I* Error Severity
IHESVAR     DS
I                                     B   1   40ERRSEV
I* Error Code
IHECVAR     DS
I                                     B   1   40ERRCOD
```

HAB A unique identifier (handle) for a PrintManager anchor block

HPRM A print session identifier

LONG A 32-bit integer in the following format:

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7.
I* Long integer
IILONG      DS
I                                     B   1   40LONG
```

OPTDATA1 This type maps to the following parameters on the SPRSOPT call:

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7.
I* Print option name input length
IOLVAR      DS
I                                     B   1   40OPTL
I* Print option name
IOPVAR      DS
I                                         1 <x> OPT
I* Value type
IVTVAR      DS
I                                     B   1   40VALTYP
I* Print option value input length
IVLVAR      DS
I                                     B   1   40VALL
I* Print option value
IVAVAR      DS
I                                         1 <x> VAL
```

On the SPRQOPT call, OPTDATA1 maps to the following (note the additional VALDL parameter):

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7...
I* Print option name input length
IOLVAR       DS
I                                      B   1   40OPTL
I* Print option name
IOPVAR       DS
I                                          1 <x> OPT
I* Value type
IVTVAR       DS
I                                      B   1   40VALTYP
I* Print option value area length
IVLVAR       DS
I                                      B   1   40VALL
I* Print option value data length
IDLVAR       DS
I                                      B   1   40VALDL
I* Print option value
IVAVAR       DS
I                                          1 <x> VAL
```

NOTE: If the print-option value is type *PBINRY*, then the data in the value parameter cannot be directly processed in RPG. Also, the *VALL* parameter on SPRSOPT must be set to the exact size of the value data since padded blanks may be invalid for a binary value.

SDF A self-defining field. SDF maps to the "continue flag" variable in RPG:

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7...
I* SDF structure
ICONVAR      DS
I                                      B   1   40CONTIN
```

This field is set to *PGETF* before the first in one or more of a series of SPRLOPT calls and is otherwise not referenced by the RPG application.

SHORT (See LONG)

STRL A string with an implicit length count.

This type maps to one of the following:

```
I*..1....+....2....+....3....+....4....+....5....+....6....+....7.
I* Input parameter
I* String input length
ILLVAR      DS
I                                       B   1   40STRL
I* String
I*
ISTRING     DS
I                                           1 <x> STRVAR


I*..1....+....2....+....3....+....4....+....5....+....6....+....7.
I* Output parameter
I* String input length
ILLVAR      DS
I                                       B   1   40STRL
I* String data length
ILDLVAR     DS
I                                       B   1   40STRDL
I* String
I*
ISTRING     DS
I                                           1 <x> STRVAR
```

The first form is used when the string is passed as an input parameter, and the second form is used when the string is an output parameter.

ULONG (See LONG)

USHORT (See LONG)

SPRABRT (Abort)

MVS	VM	OS/400	OS/2
		X	

Function

Aborts the print session.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRABRT'
C                   PARM           HPRM
C                   PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                  B   1   40HPRM
I* Success/failure return code
ISUCVAR     DS
I                                  B   1   40SUCCES
```

Figure D-1. RPG Syntax of the SPRABRT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use the SPRABRT verb to abort the print session and to delete the print job (if active). When the SPRABRT verb is issued, the *hprm* from the aborted session is no longer valid.

Ending your application without calling SPRCLOS or SPRABRT could cause unpredictable results.

Return Values

SUCCES
A nonzero value means the print session was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADOC (Abort Document)

MVS	VM	OS/400	OS/2
		X	

Function

Aborts and deletes the print job.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                    CALL 'SPRADOC'
C                    PARM           HPRM
C                    PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                   B   1   40HPRM
I* Success/failure return code
ISUCVAR     DS
I                                   B   1   40SUCCES
```

Figure D-2. RPG Syntax of SPRADOC Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use the SPRADOC verb to abort processing of the print job and to delete it. When the SPRADOC verb is issued, PrintManager returns to Session Active state.

SPRADOC does not abort a print job that has already been ended with the SPREDOC verb.

Return Values

SUCCES
A nonzero value means the print job was aborted successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRADDF (Add File)

MVS	VM	OS/400	OS/2
		X	

Function

Writes a file of print data to the print job.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRADDF'
C                   PARM           HPRM
C                   PARM           FILNML
C                   PARM           FILNAM
C                   PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                  B   1   40HPRM
I* File name field length
INMLVAR     DS
I                                  B   1   40FILNML
I* File name
INAMVAR     DS
I                                      1 <x> FILNAM
I* Success/failure return code
ISUCVAR     DS
I                                  B   1   40SUCCES
```

Figure D-3. RPG Syntax of the SPRADDF Verb

Parameters

HPRM
(Input). Identifies a valid print session.

FILNML
(Input). The length of the *FILNAM* parameter.

FILNAM
(Input). A string containing the name of a file with print data to be sent to the spool. The file name must follow the naming convention for OS/400.

Usage

Use the SPRADDF verb to write a file of print data to the print job. If an error occurs, PrintManager will return you to Session Active state with the exception of two errors: 16412 ("cannot open file") and 16547 ("read error"). These two errors will leave you in Job-in-Progress state. See "PrintManager Interface Operating States" on page 3-5 for more information on operating states.

Note: The file specified in the *FILNAM* parameter must be a physical data file with FILETYPE (*DATA) and LVLCHK (*NO).

Return Values

SUCCES

A nonzero value means the named file was written to the print job, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRCLOS (Close)

MVS	VM	OS/400	OS/2
		X	

Function

Ends the current print session.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRCLOS'
C                   PARM           HPRM
C                   PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                  B   1   40HPRM
I* Success/failure return code
IHSPSUC     DS
I                                  B   1   40SUCCES
```

Figure D-4. RPG Syntax of SPRCLOS Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use the SPRCLOS verb to end a print session and to free storage associated with the session. When SPRCLOS is issued, the *hprm* parameter from the closed session is no longer valid.

Ending your application without calling SPRCLOS or SPRABRT could cause unpredictable results.

Return Values

SUCCES
A nonzero value means the print session ended successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPREDOC (End Document)

MVS	VM	OS/400	OS/2
		X	

Function

Ends the print job.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                    CALL 'SPREDOC'
C                    PARM           HPRM
C                    PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                   B   1   40HPRM
I* Success/failure return code
ISUCVAR     DS
I                                   B   1   40SUCCES
```

Figure D-5. RPG Syntax of SPREDOC Verb

Parameters

HPRM
(Input). Identifies a valid print session.

Usage

Use SPREDOC to end a print job. When the SPREDOC verb is issued, PrintManager returns to Session Active state.

Return Values

SUCCES
A nonzero value means the print job ended successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRFERI (Free Error Information)

MVS	VM	OS/400	OS/2
		X	

Function

Frees storage associated with the error-information handle returned from the SPRGERI verb.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRFERI'
C                   PARM           ERRINF
C                   PARM           SUCCES

I* Error Information handle
IHERVAR     DS
I                                    B   1   40ERRINF
I* Success/failure return code
ISUCVAR     DS
I                                    B   1   40SUCCES
```

Figure D-6. RPG Syntax of SPRFERI Verb

Parameters

ERRINF
(Input). The **ERRINF** handle.

Usage

Use SPRFERI to free storage for the error-information handle you specify in **ERRINF**. You should issue SPRFERI for every error-information handle returned with the SPRGERI verb.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

SUCCES

A nonzero value means storage was freed successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGEEM (Get Error Message)

MVS	VM	OS/400	OS/2
		X	

Function

Gets additional error information associated with the error-information handle.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRGEEM'
C                   PARM           ERRINF
C                   PARM           INDEX
C                   PARM           MSGID
C                   PARM           ERRDL
C                   PARM           ERRMSG
C                   PARM           TTLCNT
C                   PARM           SUCCES

I* Error Information handle
IHERVAR     DS
I                                  B   1   40ERRINF
I* Index into error information
IINDVAR     DS
I                                  B   1   40INDEX
I* Message ID
IIDVAR      DS
I                                  B   1   40MSGID
I* Error message data length
IMLVAR      DS
I                                  B   1   40ERRDL
I* Error message
IERRMSG     DS
I                                      1 256 ERRMSG
I* Total number of error messages in error information
ITTLVAR     DS
I                                  B   1   40TTLCNT
I* Success/failure return code
ISUCVAR     DS
I                                  B   1   40SUCCES
```

Figure D-7. RPG Syntax of SPRGEEM Verb

Parameters

ERRINF
(Input). The error-information handle (**ERRINF**) returned by SPRGERI.

INDEX
(Input). Specifies which additional message to return. Use a value of 1 to select the first message, a value of 2 to select the second message, and so on.

MSGID
(Output). The message identifier.

ERRDL
(Output). The data length of the error message.

ERRMSG
(Output). The error message.

TTLCNT
(Output). SPRGEEM updates this parameter to specify the number of additional messages associated with the error-information handle.

Usage

The SPRGEEM verb returns additional error information in the output parameters. The original error-information handle returned from SPRGERI must be passed to SPRGEEM.

Information returned from SPRGEEM is often useful in further resolving the cause of errors. For example, for error code 16412, the *ERRMSG* parameter provides the name of the file where the error occurred. For operating system errors, PrintManager updates the *ERRMSG* parameter with error text provided by the operating system. Information in the *ERRMSG* parameter is truncated to 256 characters, if necessary.

There may be 1 to *n* additional error messages associated with the error information handle. Set *INDEX* to 0 to get the total number of messages that can be returned. If the value returned in *TTLCNT* is 0, there are no additional messages. If there are additional messages, set *INDEX* to correspond to the message you want to return. If you want to return all additional messages, call SPRGEEM repeatedly, incrementing *INDEX* until *TTLCNT* is reached.

Note: Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

SUCCES
A nonzero value means the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRGERI (Get Error Information)

MVS	VM	OS/400	OS/2
		X	

Function

Gets error information from the last error.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRGERI'
C                   PARM           HAB
C                   PARM           ERRSEV
C                   PARM           ERRCOD
C                   PARM           ERRINF

I* Application Anchor Block handle
IHABVAR     DS
I                                  B   1   40HAB
I* Error Severity
IHESVAR     DS
I                                  B   1   40ERRSEV
I* Error Code
IHECVAR     DS
I                                  B   1   40ERRCOD
I* Error Information handle
IHERVAR     DS
I                                  B   1   40ERRINF
```

Figure D-8. RPG Syntax of SPRGERI Verb

Parameters

HAB
(Input). The anchor-block handle returned from a successful call to SPRINIT.

ERRSEV
(Output). Error severity.

ERRCOD
(Output). Error code.

ERRINF
(Output). The **ERRINF** handle.

Usage

Use the SPRGERI verb to get error information associated with the last PrintManager Interface error. You can then use the error-information handle (**ERRINF**) as input to the SPRGEEM verb to get additional information (if any) about the error. To free storage for the error-information handle, use SPRFERI.

Notes:

1. If PrintManager cannot be initialized (with SPRINIT (Initialize PrintManager)), SPRGERI (Get Error Information) will not return an error-information buffer.
2. Error information is not cleared (another call to SPRGERI (Get Error Information) will return the same information). Also, error information is not cleared by subsequent calls to SPRGERI (Get Error Information) or SPRFERI (Free Error Information).

Return Values

ERRINF

The SPRGERI verb returns an error-information handle structure for the last error. This handle can be used with the SPRGEEM (Get Error Message) verb to obtain more information about the error. See Appendix F, "Verb Error Codes" on page F-1 for information on PrintManager Interface verb error codes. A return value of 0 indicates that no previous error occurred.

As well as errors from PrintManager Interface verbs, you can get errors from API verbs because some PrintManager Interface verbs invoke PrintManager API verbs.

Environment Restrictions

None.

SPRINIT (Initialize PrintManager)

MVS	VM	OS/400	OS/2
		X	

Function

Initialize PrintManager.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRINIT'
C                   PARM           OPTION
C                   PARM           HAB

I* Initialization options
IOPTVAR     DS
I                                  B   1   400PTION
I* Application Anchor Block handle
IHABVAR     DS
I                                  B   1   40HAB
```

Figure D-9. RPG Syntax of SPRINIT Verb

Parameters

OPTION
(Input). This parameter is reserved and should have a value of 0.

Usage

Use the SPRINIT verb to initialize PrintManager. You cannot issue any other PrintManager verbs (including error verbs) until you have successfully initialized PrintManager.

Return Values

HAB
The SPRINIT verb should return an anchor-block handle (HAB). A return value of 0, however, indicates an error while initializing PrintManager.

Ensure that you save the HAB value that the SPRINIT verb returns. You will need to specify this value on other PrintManager verbs. You cannot issue two successive calls to SPRINIT. You must terminate PrintManager with the SPRTERM verb before reinitializing PrintManager.

Environment Restrictions

None.

SPRLOPT (List Options)

MVS	VM	OS/400	OS/2
		X	

Function

List current print options for a print session.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRLOPT'
C                   PARM           HPRM
C                   PARM           COUNT
C                   PARM           OPTARY
C                   PARM           ITMRET
C                   PARM           ITMREM
C                   PARM           CONTIN
C                   PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                  B   1   40HPRM
I* Number of elements in OPTARY
ICNTVAR     DS
I                                  B   1   40COUNT
I* Option names (a multiple occurrence data structure)
IOPTARY                             <x>
I*   Option name
I                                      1  31 OPTNAM
I* Number of items returned
IRETVAR     DS
I                                  B   1   40ITMRET
I* Number of items remaining
IREMVAR     DS
I                                  B   1   40ITMREM
I* Continue flag
ICONVAR     DS
I                                  B   1   40CONTIN
I* Success/failure return code
ISUCVAR      DS
I                                  B   1   40SUCCES
```

Figure D-10. RPG Syntax of SPRLOPT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

COUNT
(Input). Specifies the number of elements in *OPTARY*.

OPTARY
(Output). Returns a list of current print-option names.

ITMRET
(Output). Indicates the number of print-option names returned in *OPTARY*.

ITMREM
(Output). For a partial list, indicates the number of option names not yet listed; otherwise, *ITMREM* is 0.

CONTIN
(Input/Output). Used to get a list of options in a series of calls (see "Usage"). If you are not using this field for a series of calls, store **PGETF** in it.

Usage

Use the SPRLOPT verb to list the set of current options in a print session.

If you don't know the maximum number of current options that you will have in a print session, you can list all the options by using the *CONTIN* flag as follows:

- Before the first call to SPRLOPT, set *CONTIN* to **PGETF**. If all of the options will not fit in the option array, then *ITMREM* will be greater than zero, indicating that there are more names in the current options.
- Subsequent calls to SPRLOPT will result in more option names being returned in your option array. When *ITMREM* is zero, all of the current options were returned in the option array.

  Note: Any changes to the print options with SPRLOPT during a series of calls using *CONTIN* will not be reflected in the partial lists returned.

Return Values

SUCCES
A nonzero value means the list request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPROPEN (Open)

MVS	VM	OS/400	OS/2
		X	

Function

Opens and identifies a print session.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPROPEN'
C                   PARM           HAB
C                   PARM           PRDNL
C                   PARM           PRDNAM
C                   PARM           BUFRL
C                   PARM           BUFR
C                   PARM           HPRM

I* Application Anchor Block handle
IHABVAR     DS
I                                  B   1   40HAB
I* Prd Name field length
ILENVAR     DS
I                                  B   1   40PRDNL
I* Prd Name
INAMVAR     DS
I                                      1 <x> PRDNAM
I* Buffer field length (must be 0; BUFR reserved for future use)
IBFLVAR     DS
I                                  B   1   40BUFRL
I* Buffer (reserved for future use)
IBUFVAR     DS
I                                      1   2 BUFR
I* Handle for Print session
IHSPVAR     DS
I                                  B   1   40HPRM
```

Figure D-11. RPG Syntax of SPROPEN Verb

Parameters

HAB
(Input). The anchor-block handle returned from a successful call to SPRINIT.

PRDNL
(Input). The length of the *PRDNAM* parameter.

PRDNAM
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored and no validation is done if the value is all blanks, a single asterisk (*), or *PRDNL* is zero.

For information on naming conventions, refer to *PrintManager Application Programming Interface Reference*.

BUFRL
(Input). This parameter is reserved and must have a value of zero.

BUFR
(Input). This parameter is reserved.

Usage

Use the SPROPEN verb to start a print session. You can also use SPROPEN to define the set of current options by specifying a print descriptor.

Return Values

HPRM
The SPROPEN verb should return the session identifier (*hprm*). A return value of 0, however, means an error occurred. Ensure that you save the *hprm* value that the SPROPEN verb returns. You will need to specify this value on other PrintManager Interface verbs used in a session.

Environment Restrictions

None.

SPRQOPT (Query Option)

MVS	VM	OS/400	OS/2
		X	

Function

Query the current value of a print option.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRQOPT'
C                   PARM           HPRM
C                   PARM           OPTL
C                   PARM           OPT
C                   PARM           VALTYP
C                   PARM           VALAL
C                   PARM           VALDL
C                   PARM           VAL
C                   PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                  B   1   40HPRM
I* Print option name input length
IOLVAR      DS
I                                  B   1   40OPTL
I* Print option name
IOPTDAT     DS
I                                      1 <x> OPT
I* Value type
IVTVAR      DS
I                                  B   1   40VALTYP
I* Print option value area length
IALVAR      DS
I                                  B   1   40VALAL
I* Print option value data length
IDLVAR      DS
I                                  B   1   40VALDL
I* Print option value
IVAVAR      DS
I                                      1 <x> VAL
I* Success/failure return code
ISUCVAR     DS
I                                  B   1   40SUCCES
```

Figure D-12. RPG Syntax of SPRQOPT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

OPTL
(Input). Specify the print-option name input length, which can be up to 31 characters, in the *OPT* parameter. Any additional characters will be truncated.

OPT
(Input) Specify the print-option name.

VALTYP
(Output). The value type. If the value of *VALTYP* is **PSTRL**, the data is a character string. If the value of *VALTYP* is **PBINRY**, the data is binary data with a length of *VALDL*.

VALAL
(Input). The print-option value area length.

VALDL
(Output). The length of data in the *VAL* parameter.

VAL
(Output). The print-option value.

Usage

Use the SPRQOPT verb to list the current values for a print option.

If it is determined from the *VALAL* parameter that *VAL* is not large enough to hold the current values for the specified print option, then the verb will fail. *VALDL* will be set to the total length of the actual option values, and as many values as possible is returned in *VAL*. The error code will be set to X'40AA' ("buffer too small"). Therefore, *VALDL* will be greater than the value you specified in *VALAL*. To correct this, make *VAL* larger and set *VALAL* to this larger value.

See "Length Parameters" on page D-1 for a general description of how length parameters are used.

Return Values

success
A nonzero value means the request was successful, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRSDOC (Start Document)

MVS	VM	OS/400	OS/2
		X	

Function

Starts a print job and specifies a document name for the print job.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                    CALL 'SPRSDOC'
C                    PARM           HPRM
C                    PARM           DOCNML
C                    PARM           DOCNME
C                    PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                   B   1   40HPRM
I* Document Name field length
INMLVAR     DS
I                                   B   1   40DOCNML
I* Document Name
INMEVAR     DS
I                                       1 <x> DOCNME
I* Sucess/failure return code
ISUCVAR     DS
I                                   B   1   40SUCCES
```

Figure D-13. RPG Syntax of SPRSDOC Verb

Parameters

HPRM
(Input). Identifies a valid print session.

DOCNML
(Input). Specify the *DOCNME* input length.

DOCNME
(Input). A string containing the document name for the print job. If no document name is specified, the name is inherited from the previous print job in a session. If no previous document name exists, the operating system provides a default. The *DOCNME* parameter must follow OS/400 naming conventions, and is truncated to 10 characters.

Usage

Use SPRSDOC to start a print job. You must issue SPREDOC before issuing another SPRSDOC within the same print session.

If the SPRSDOC verb is successful, PrintManager is placed in Job-in-Progress state.

Return Values

SUCCES

A nonzero value means a new print job was initiated, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRSOPT (Set Option)

MVS	VM	OS/400	OS/2
		X	

Function

Set or change a print option.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                    CALL 'SPRSOPT'
C                    PARM           HPRM
C                    PARM           PRDNL
C                    PARM           PRDNAM
C                    PARM           OPTL
C                    PARM           OPT
C                    PARM           VALTYP
C                    PARM           VALL
C                    PARM           VAL
C                    PARM           SUCCES

I* Handle for Print session
IHSPVAR      DS
I                                   B   1   40HPRM
I* Prd Name field length
ILENVAR      DS
I                                   B   1   40PRDNL
I* Prd Name
INAMVAR      DS
I                                       1 <x> PRDNAM
I* Print option name input length
IOLVAR       DS
I                                   B   1   40OPTL
I* Print option name
IOPTDAT      DS
I                                       1 <x> OPT
I* Value type
IVTVAR       DS
I                                   B   1   40VALTYP
I* Print option value input length
IVLVAR       DS
I                                   B   1   40VALL
I* Print option value
IVAVAR       DS
I                                       1 <x> VAL
I* Success/failure return code
ISUCVAR      DS
I                                       1   40SUCCES
```

Figure D-14. RPG Syntax of SPRSOPT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

PRDNL
(Input). The length of the *PRDNAM* parameter.

PRDNAM
(Input). The name of the print descriptor to be used for print-option value validation and default values. This value is ignored if it is all blanks, or a single asterisk (*), or the *PRDNAM* parameter is zero. If *DEL is specified, all print options are deleted from the set of current options, the print descriptor is no longer in effect, and any option specified in the other parameters is not validated.

OPTL
(Input). Specify the print-option name input length, which can be up to 31 characters, in the *OPT* parameter. Any additional characters will be truncated.

OPT
(Input) Specify the print-option name.

VALTYP
(Input). The value type. If the value of *VALTYP* is **PSTRL**, the data is a character string. If the value of *VALTYP* is **PBINRY**, the data is binary data with a length of *VALL*.

VALL
(Input). The length of data in the *VAL* parameter.

VAL
(Input). The print-option value.

Usage

Use the SPRSOPT verb to validate and set print-option information and to specify a print descriptor (see "Print Option Validation" on page 5-5).

If an error occurs in retrieving a print descriptor, any defaults and validation information from the previous print descriptor are no longer in effect.

If the *PRDNAM* parameter is specified as all blanks, an asterisk, or the *PRDNAM* parameter is zero, the options specified on this call are validated against the current print descriptor, if one has been previously defined. If no print descriptor has been defined, no validation is performed.

If a print descriptor is specified after print options have been set, the previously set print options are validated against the new print descriptor.

You should specify a print descriptor only once because each time you specify a print descriptor, it is reapplied and the set of current options is validated.

Return Values

SUCCES

A nonzero value means the print-option values were set successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRTERM (Terminate PrintManager)

MVS	VM	OS/400	OS/2
		X	

Function

Terminate PrintManager.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                   CALL 'SPRTERM'
C                   PARM           HAB
C                   PARM           SUCCES

I* Application Anchor Block handle
IHABVAR     DS
I                                  B   1   40HAB
I* Success/failure return code
ISUCVAR     DS
I                                  B   1   40SUCCES
```

Figure D-15. RPG Syntax of SPRTERM Verb

Parameters

HAB
(Input). The anchor-block handle returned from a successful call to SPRINIT.

Usage

Use the SPRTERM verb to terminate PrintManager and to release all associated resources. For the *hab* parameter, ensure that you enter the anchor-block handle returned on the SPRINIT verb when you initialized PrintManager. You must terminate a PrintManager session before you can reinitialize PrintManager.

Return Values

SUCCES
A nonzero value means PrintManager was terminated successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

SPRWRIT (Write)

MVS	VM	OS/400	OS/2
		X	

Function

Sends a buffer of data to the print job.

Syntax

```
C*..1....+....2....+....3....+....4....+....5....+....6....+....7...
C                    CALL 'SPRWRIT'
C                    PARM           HPRM
C                    PARM           BUFFL
C                    PARM           BUFFER
C                    PARM           SUCCES

I* Handle for Print session
IHSPVAR     DS
I                                   B   1   40HPRM
I* BUFFER input length (length of data to be written)
IBFLVAR     DS
I                                   B   1   40BUFFL
I* Data to be written
IBUFVAR     DS
I                                       1 <x> BUFFER
I* Success/failure return code
ISUCVAR     DS
I                                   B   1   40SUCCES
```

Figure D-16. RPG Syntax of SPRWRIT Verb

Parameters

HPRM
(Input). Identifies a valid print session.

BUFFL
(Input). The length of the buffer of print data sent to the print job.

BUFFER
(Input). The buffer of print data sent to the print job.

Usage

Use the SPRWRIT verb to send a buffer of data to a print job. If an error occurs, PrintManager returns to Session Active state.

Return Values

SUCCES

A nonzero value means the buffer of data was sent to the print job successfully, whereas a value of 0 means an error occurred.

Environment Restrictions

None.

Example of a PrintManager Session in RPG Language

```
*****************************************************************
*      PRINTMANAGER INTERFACE EXAMPLE --- RPG                   *
*                                                               *
* This example shows how to use the PrintManager Print          *
* Interface. It is not intended to satisfy any particular       *
* functional needs.                                             *
*                                                               *
* Sequence of Operations:                                       *
*                                                               *
* - Start a PrintManager process                      (SPRINIT) *
* - Open a print session and set a print descriptor    (SPROPEN) *
* - Set a valid and an invalid print option            (SPRSOPT) *
* - Get a list of all current print options            (SPRLOPT) *
* - Query each print option and print the value        (SPRQOPT) *
* - Start the print job                                (SPRSDOC) *
* - Write buffers of data to the print job             (SPRWRIT) *
* - End the print job.                                 (SPREDOC) *
* - Close the print session                            (SPRCLOS) *
* - Terminate the PrintManager process                 (SPRTERM) *
*                                                               *
*****************************************************************
```

Figure D-17 (Part 1 of 16). Example of a PrintManager Print Session in RPG Language

```
     *
     ****************************************************************
     *                                                              *
     *                  FILE SPECIFICATIONS                         *
     *                                                              *
     ****************************************************************
     *..1....+....2....+....3....+....4....+....5....+....6....+....7...
     *
     *  Define output file
     *
    FQPRINT  O   F     132     OF     PRINTER      KPRTCTLLINE
     ****************************************************************
     *                                                              *
     *                  INPUT SPECIFICATIONS                        *
     *                                                              *
     ****************************************************************
     *..1....+....2....+....3....+....4....+....5....+....6....+....7...
     *--------------------------------------------------------------*
     * IBM-supplied constants                                       *
     *--------------------------------------------------------------*
    I/COPY EKICONR
     *--------------------------------------------------------------*
     * Print descriptor name.                                       *
     *                                                              *
     * In this example note that the "universal" print descriptor   *
     * name format is used.  This is to make the code as system-    *
     * independent (and portable) as possible.                      *
     *                                                              *
     * This example also assumes that this print descriptor already *
     * exists and that it contains the following print options:     *
     *                                                              *
     *      Option Name: DUPLEX                                     *
     *          Default:      YES                                   *
     *          Rule:         List                                  *
     *          Valid Values: YES, NO, TUMBLE                       *
     *                                                              *
     *      Option Name: COPIES                                     *
     *          Default:      1                                     *
     *          Rule:         Range                                 *
     *          Valid Values: 1-255, zero decimal places            *
     *--------------------------------------------------------------*
    I             'PRDNAME=IBM Print Ma-C         CPRDNM
    I             'nager Options'
```

Figure D-17 (Part 2 of 16). Example of a PrintManager Print Session in RPG Language

```
 *-------------------------------------------------------------*
 * General Declarations                                        *
 *-------------------------------------------------------------*
IGENVAR      DS
 *   Success/failure return code
I                                     B   1   40SUCCES
 *   General counter
I                                     B   5   80COUNT
 *   Level of structure
I                                     B   9  120LEVEL
 *   Loop index
I                                     B  13  160I
 *   Initialization options
I                                     B  17  200OPTS
 *   Application Anchor Block handle
I                                     B  21  240HAB
 *   Handle for Print Session
I                                     B  29  320HPRM
 *   Number of Items Returned
I                                     B  33  360ITMRTN
 *   Number of Items Remaining
I                                     B  37  400ITMRMN
 *   Number of elements in OPTARY
I                                     B  49  520OPTACT
 *   Name of Print Descriptor
IPRDEVR      DS
I                                     B   1   40PRDEL
I                                         5  84 PRDENM
 *   Buffer (reserved for future use)
IBFRVAR      DS
I                                     B   1   40BUFRL
I                                         5   6 BUFR
 *   Write Buffer
IWBUFF       DS
I                                         1 132 BUFF
 *   Document length and name
IDOCVAR      DS
I                                     B   1   40DOCNML
I                                         5  14 DOCNME
```

Figure D-17 (Part 3 of 16). Example of a PrintManager Print Session in RPG Language

```
 *-------------------------------------------------------------*
 * PrintManager Structure Definitions                          *
 *-------------------------------------------------------------*
 *   Option Data variables
IOPTVAR      DS
I                                      B   1   40OPTL
I                                          5  36 OPT
I                                      B  37  400VALTYP
I                                      B  41  440VALL
I                                      B  45  480VALDL
I                                         49 128 VAL
 *   Continue flag
ICONVAR      DS
I                                      B   1   40CONTIN
 *   Option Name Array Structure
IOPTARY      DS                           20
 *     Option name
I                                          1  31 OPTNAM
 *-------------------------------------------------------------*
 * Declares for data written to the print job.                 *
 *-------------------------------------------------------------*
I              'First buffer of data-C         DATA1
I              ' written using SPRWR-
I              'IT'
I              'Second buffer of dat-C         DATA2
I              'a written using SPRW-
I              'RIT'
 *-------------------------------------------------------------*
 *  Variables for handling error conditions                    *
 *-------------------------------------------------------------*
IERRVAR      DS
I                                      B   1   40ERRINF
I                                      B   5   80INDX
I                                      B   9  120TOTCNT
IERRCVR      DS
I                                      B   1   40ERRCOD
I                                      B   5   80ERRSEV
IERRM        DS
I                                      B   1   40ERRID
I                                      B   5   80ERRDL
I                                          9 264 ERRMSG
 * Alternate error message for output on multiple lines
IERMM2       DS
I                                          1 132 ERMM2A
I                                        133 256 ERMM2B
 * Function name
IFNMVAR      DS
I                                          1   8 FNM
```

Figure D-17 (Part 4 of 16). Example of a PrintManager Print Session in RPG Language

```
 *----------------------------------------------------------------*
 *  Output Variables                                              *
 *----------------------------------------------------------------*
ILINE        DS                                  9
I                                              1   1 SPCBFR
I                                              3   4 SKPBFR
IACTDAT      DS
I                                              1 132 OFLD
 * Output error information structure 1
IMSGSTR      DS
I                                              1  15 MFLD1
I                                             16  30 MFLD2
I                                             31  45 MFLD3
I                                             46  60 MFLD4
I                                             61 132 MREST
 * Output error information structure 2
IMSGSTB      DS
I                                              1  15 MFLDB1
I                                             16 132 MFLDB2
I              'Error Severity:'      C          MERSEV
I              'Error Code:'          C          MERCOD
I              'Error in:'            C          MERFNM
I              'Error ID:'            C          MERID
I              'Error Message:'       C          MERMSG
 * Output print option structure
IPOPSTR      DS
I                                              1  15 PFLDA1
I                                             15 132 PFLDA2
I              'Option name:'         C          POPNAM
I              'Option value:'        C          POPVAL
 *----------------------------------------------------------------*
```

Figure D-17 (Part 5 of 16). Example of a PrintManager Print Session in RPG Language

```
 *  Miscellaneous Constants                                          *
 *-----------------------------------------------------------------*
I             'YES, NO, TUMBLE'     C         DUPVVA
I             '0, 1, 255'           C         CPSVVA
I             'Setting an invalid p-C         INVM1
I             'rint option...'
I             'An error message sho-C         INVM2
I             'uld be output...'
I             'List of current prin-C         CURPOP
I             't options:'
 ******************************************************************
 *                                                                *
 *                  CALCULATION SPECIFICATIONS                    *
 *                                                                *
 ******************************************************************
 *..1....+....2....+....3....+....4....+....5....+....6....+....7...
 *================================================================*
 * Start a PrintManager process                          (SPRINIT) *
 *================================================================*
C                     Z-ADD0         OPTS
C                     MOVE HNULL     HAB
C                     CALL 'SPRINIT'
C                     PARM           OPTS
C                     PARM           HAB
C                     MOVE HAB       SUCCES
C                     MOVE 'SPRINIT 'FNM
C                     EXSR CHKNUM
```

Figure D-17 (Part 6 of 16). Example of a PrintManager Print Session in RPG Language

```
 *===============================================================*
 * Open a print session and set a print descriptor     (SPROPEN) *
 *===============================================================*
C                     Z-ADD0         BUFRL
 * Set the Print Descriptor name and lengths
C                     MOVE *BLANK    PRDENM
C                     MOVELCPRDNM    PRDENM
C                     Z-ADD80        PRDEL
C                     CALL 'SPROPEN'
C                     PARM           HAB
C                     PARM           PRDEL
C                     PARM           PRDENM
C                     PARM           BUFRL
C                     PARM           BUFR
C                     PARM           HPRM
C                     MOVE HPRM      SUCCES
C                     MOVE 'SPROPEN 'FNM
C                     EXSR CHKNUM
 *===============================================================*
 * Set a valid print option                           (SPRSOPT) *
 *===============================================================*
 *
 *---------------------------------------------------------------*
 * Define buffer level as using an OPTDATA1 structure and define *
 * constant values used in OPTDATA1 structure for setting        *
 * options.                                                      *
 *---------------------------------------------------------------*
C                     Z-ADD32        OPTL
C                     Z-ADD80        VALL
C                     MOVE PSTRL     VALTYP
```

Figure D-17 (Part 7 of 16). Example of a PrintManager Print Session in RPG Language

```
 *-------------------------------------------------------------*
 * Set DUPLEX option ...                                       *
 *-------------------------------------------------------------*
 * No new Print Descriptor
C                    MOVEL'*'       PRDENM
C                    Z-ADD1         PRDEL
 * Option name = DUPLEX
C                    MOVE *BLANK    OPT
C                    MOVEL'DUPLEX'  OPT
 * Option value = YES
C                    MOVE *BLANK    VAL
C                    MOVEL'YES'     VAL
 * Set the DUPLEX option
C                    CALL 'SPRSOPT'
C                    PARM           HPRM
C                    PARM           PRDEL
C                    PARM           PRDENM
C                    PARM           OPTL
C                    PARM           OPT
C                    PARM           VALTYP
C                    PARM           VALL
C                    PARM           VAL
C                    PARM           SUCCES
C                    MOVE 'SPRSOPT 'FNM
C                    EXSR CHKNUM
```

Figure D-17 (Part 8 of 16). Example of a PrintManager Print Session in RPG Language

```
 *============================================================*
 * Set an invalid print option                      (SPRSOPT) *
 *============================================================*
 *
 * We're setting an invalid print option, so an error message
 * should be output.
 *
C                    MOVE *BLANK    OFLD
C                    MOVELINVM1     OFLD
C                    EXCPTPRINT                      Print
C                    MOVE *BLANK    OFLD
C                    MOVELINVM2     OFLD
C                    EXCPTPRINT                      Print
 *------------------------------------------------------------*
 * Set COPIES option to an invalid value ...                  *
 * Since the valid range in the print descriptor is 1 to 255,  *
 * 300 is invalid and the value of COPIES will remain the      *
 * default value from the print descriptor which is 1.         *
 *------------------------------------------------------------*
 * Option name = COPIES
C                    MOVE *BLANK    OPT
C                    MOVEL'COPIES'  OPT
 * Option value = 300
C                    MOVE *BLANK    VAL
C                    MOVEL'300'     VAL
 * Set the COPIES option
C                    CALL 'SPRSOPT'
C                    PARM           HPRM
C                    PARM           PRDEL
C                    PARM           PRDENM
C                    PARM           OPTL
C                    PARM           OPT
C                    PARM           VALTYP
C                    PARM           VALL
C                    PARM           VAL
C                    PARM           SUCCES
C                    MOVE 'SPRSOPT 'FNM
C                    EXSR CHKNUM
```

Figure D-17 (Part 9 of 16). Example of a PrintManager Print Session in RPG Language

```
 *=================================================================*
 * List and query each print option.                               *
 * Output the results of each query.  This is done as an outer     *
 * loop to continue listing option names as long as there are      *
 * any left, and an inner loop to query and output each option     *
 * in the current list.                                            *
 *=================================================================*
 * Output "header" message
C                    MOVE *BLANK    OFLD
C                    EXCPTPRINT                      Print
C                    MOVELCURPOP    OFLD
C                    EXCPTPRINT                      Print
 * Initialize for the loops
C                    MOVE PGETF     CONTIN
C                    Z-ADD20        OPTACT
 *
 * Do until there are no more print options to list...
 *
C           ITMRMN   DOUEQ0
 *=================================================================*
 * List Options                                          (SPRLOPT) *
 *=================================================================*
C                    CALL 'SPRLOPT'
C                    PARM           HPRM
C                    PARM           OPTACT
C                    PARM           OPTARY
C                    PARM           ITMRTN
C                    PARM           ITMRMN
C                    PARM           CONTIN
C                    PARM           SUCCES
C                    MOVE 'SPRLOPT 'FNM
C                    EXSR CHKNUM
 *
 * Do for each of the print options in the list...
 *
C           1        DO   ITMRTN    INDX
 * Set up for query
C           INDX     OCUR OPTARY
C                    MOVE *BLANK    OPT
C                    MOVE OPTNAM    OPT
 *=================================================================*
 * Query Option                                          (SPRQOPT) *
 *=================================================================*
C                    CALL 'SPRQOPT'
C                    PARM           HPRM
C                    PARM           OPTL
C                    PARM           OPT
C                    PARM           VALTYP
C                    PARM           VALL
C                    PARM           VALDL
C                    PARM           VAL
C                    PARM           SUCCES
C                    MOVE 'SPRQOPT 'FNM
C                    EXSR CHKNUM
```

Figure D-17 (Part 10 of 16). Example of a PrintManager Print Session in RPG Language

```
     *-------------------------------------------------------------*
     * Output the print option                                     *
     *-------------------------------------------------------------*
     * Output the option name
    C                   MOVE *BLANK    POPSTR
    C                   MOVELPOPNAM    PFLDA1
    C                   MOVELOPT       PFLDA2
    C                   MOVELPOPSTR    OFLD
    C                   EXCPTPRINT                    Print
     * Output the option value
    C                   MOVE *BLANK    POPSTR
    C                   MOVELPOPVAL    PFLDA1
    C                   MOVELPOPSTR    OFLD
    C                   EXCPTPRINT                    Print
    C                   MOVE *BLANK    OFLD
    C                   MOVELVAL       OFLD
    C                   EXCPTPRINT                    Print
     * End of inner iterative loop
    C                   END
     * End of outer DO UNTIL loop
    C                   END
```

Figure D-17 (Part 11 of 16). Example of a PrintManager Print Session in RPG Language

```
 *===============================================================*
 * Start the print job                                (SPRSDOC) *
 *===============================================================*
C                    Z-ADD7         DOCNML
C                    MOVEL'EXAMPLE' DOCNME
C                    CALL 'SPRSDOC'
C                    PARM           HPRM
C                    PARM           DOCNML
C                    PARM           DOCNME
C                    PARM           SUCCES
C                    MOVE 'SPRSDOC 'FNM
C                    EXSR CHKNUM
 *===============================================================*
 * Write buffers of data to the print job             (SPRWRIT) *
 *===============================================================*
 * Write the first buffer of data
C                    Z-ADD42        COUNT
C                    MOVELDATA1     BUFF
C                    CALL 'SPRWRIT'
C                    PARM           HPRM
C                    PARM           COUNT
C                    PARM           BUFF
C                    PARM           SUCCES
C                    MOVE 'SPRWRIT 'FNM
C                    EXSR CHKNUM
 * Write the second buffer of data
C                    Z-ADD43        COUNT
C                    MOVELDATA2     BUFF
C                    CALL 'SPRWRIT'
C                    PARM           HPRM
C                    PARM           COUNT
C                    PARM           BUFF
C                    PARM           SUCCES
C                    MOVE 'SPRWRIT 'FNM
C                    EXSR CHKNUM
```

Figure D-17 (Part 12 of 16). Example of a PrintManager Print Session in RPG Language

```
 *===============================================================*
 * End the print job                                  (SPREDOC) *
 *===============================================================*
C                    CALL 'SPREDOC'
C                    PARM           HPRM
C                    PARM           SUCCES
C                    MOVE 'SPREDOC 'FNM
C                    EXSR CHKNUM
 *===============================================================*
 * Close the print session                            (SPRCLOS) *
 *===============================================================*
C                    CALL 'SPRCLOS'
C                    PARM           HPRM
C                    PARM           SUCCES
C                    MOVE 'SPRCLOS 'FNM
C                    EXSR CHKNUM
 *===============================================================*
 * Terminate the PrintManager process                 (SPRTERM) *
 *===============================================================*
C                    CALL 'SPRTERM'
C                    PARM           HAB
C                    PARM           SUCCES
C                    MOVE 'SPRTERM 'FNM
C                    EXSR CHKNUM
 *
 * End of Main-line code
 *
C                    SETON                      LR
 ***************************************************************
 *                    Check Error Function                     *
 *                                                             *
 * The input to this subroutine is the SUCCES variable.        *
 * This subroutine checks the content of SUCCES to see if an   *
 * error occurred.  If an error did occur, the name of the     *
 * function (FNM) which caused the error is output, along with *
 * any additional error information obtained.                  *
 ***************************************************************
 *
 * Check for success/failure
 *
C           CHKNUM    BEGSR
C           SUCCES    IFEQ PFALSE
 *---------------------------------------------------------------*
 * Error occurred - gather error information to output         *
 *---------------------------------------------------------------*
 * Output the name of the function which had the error
C                    MOVE *BLANK    MSGSTR
C                    MOVE *BLANK    OFLD
C                    MOVELMERFNM    MFLD1
C                    MOVELFNM       MFLD2
C                    MOVE MSGSTR    OFLD
C                    EXCPTPRINT                    Print
```

Figure D-17 (Part 13 of 16). Example of a PrintManager Print Session in RPG Language

```
     *===============================================================*
     * Display the error information                      (SPRGERI) *
     *                                                    (SPRGEEM) *
     *===============================================================*
     *
     *---------------------------------------------------------------*
     * Get the error code and the error information handle (SPRGERI) *
     *---------------------------------------------------------------*
    C                   Z-ADD0         ERRINF
    C                   CALL 'SPRGERI'
    C                   PARM           HAB
    C                   PARM           ERRSEV
    C                   PARM           ERRCOD
    C                   PARM           ERRINF
     * If we were unable to get error information, then just exit
    C           ERRINF  IFEQ 0000000000
    C                   GOTO ENDIT
    C                   END
     * Output the error severity and error code.
     * Fill in the output fields and output the information.
    C                   MOVE *BLANK    MSGSTR
    C                   MOVE *BLANK    OFLD
    C                   MOVE MERCOD    MFLD1
    C                   MOVE ERRCOD    MFLD2
    C                   MOVE MERSEV    MFLD3
    C                   MOVE ERRSEV    MFLD4
    C                   MOVELMSGSTR    OFLD
    C                   EXCPTPRINT                      Print
```

Figure D-17 (Part 14 of 16). Example of a PrintManager Print Session in RPG Language

```
     *-------------------------------------------------------------*
     * Invoke SPRGEEM with an index value of 0 to get the total    *
     * number of additional error messages.                        *
     *-------------------------------------------------------------*
    C                     Z-ADD0         INDX
    C                     CALL 'SPRGEEM'
    C                     PARM           ERRINF
    C                     PARM           INDX
    C                     PARM           ERRID
    C                     PARM           ERRDL
    C                     PARM           ERRMSG
    C                     PARM           TOTCNT
    C                     PARM           SUCCES
     *-------------------------------------------------------------*
     * Output the error identifier and error value for each        *
     * additional error message.                                   *
     *-------------------------------------------------------------*
     * DO 1 to TOTCNT times...
    C           1         DO   TOTCNT    INDX
     *
     * Get the additional error message
     *
    C                     CALL 'SPRGEEM'
    C                     PARM           ERRINF
    C                     PARM           INDX
    C                     PARM           ERRID
    C                     PARM           ERRDL
    C                     PARM           ERRMSG
    C                     PARM           TOTCNT
    C                     PARM           SUCCES
     * If didn't get the error message, don't try to output it
    C           SUCCES    IFEQ 0000000000
    C                     GOTO SKPOUT
    C                     END
     * Fill in the output fields and output the information
     *
     * Output the constant line saying "Error Message:"
    C                     MOVE *BLANK    MSGSTR
    C                     MOVE *BLANK    OFLD
    C                     MOVE ERRID     MFLD1
    C                     MOVELMSGSTR    OFLD
    C                     EXCPTPRINT                      Print
```

Figure D-17 (Part 15 of 16). Example of a PrintManager Print Session in RPG Language

```
 * Move the error message into a structure such that we can
 * output it on multiple lines
C                    MOVE ERRMSG    ERMM2
 * Output the first 132 characters of the message
C                    MOVE *BLANK    OFLD
C                    MOVE ERMM2A    OFLD
C                    EXCPTPRINT                      Print
 * Output the next 124 characters of the message
C                    MOVE *BLANK    OFLD
C                    MOVE ERMM2B    OFLD
C                    EXCPTPRINT                      Print
C           SKPOUT   TAG
 * End DO 1 to TOTCNT times
C                    END
 *---------------------------------------------------------------*
 * The storage for the ERRINFO structure returned by SPRGERI     *
 * must be freed using SPRFERI.                                  *
 *---------------------------------------------------------------*
C                    CALL 'SPRFERI'
C                    PARM           ERRINF
C                    PARM           SUCCES
 * End "IF not successful"
C                    END
 * Return to the caller
C           ENDIT    TAG
C                    ENDSR
 ****************************************************************
 *                                                              *
 *                 OUTPUT SPECIFICATIONS                        *
 *                                                              *
 ****************************************************************
 *..1....+....2....+....3....+....4....+....5....+....6....+....7...
OQPRINT  E               PRINT
O                        OFLD   B0132
```

Figure D-17 (Part 16 of 16). Example of a PrintManager Print Session in RPG Language

Appendix E. System-Specific Information

This appendix describes unique implementation of the PrintManager Interface in the MVS, VM, and OS/400 operating-system environments. The level of support is determined by how the printer driver and the environment handle print options.

See Table 5-2 on page 5-4 for a list of print options used when you request to build a FORMDEF using *Build.

PrintManager does not support direct-mode printing.

Initializing and Terminating PrintManager

In the MVS, VM, and OS/400 environments, use the SPRINIT (Initialize PrintManager) verb to initialize PrintManager and use the SPRTERM (Terminate PrintManager) verb to terminate PrintManager.

SPRINIT (Initialize PrintManager)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Initialize PrintManager.

Syntax

```
SPRINIT (Options, hab)
```

Figure E-1. Format of the SPRINIT (Initialize PrintManager) Verb

Parameters

Options (USHORT)
(Input). This parameter is reserved and should have a value of 0.

Usage

Use the SPRINIT (Initialize PrintManager) verb to initialize PrintManager. You cannot issue any other API verbs (including error verbs) until you have successfully initialized PrintManager.

Return Values

hab (HAB)
The SPRINIT (Initialize PrintManager) verb should return an anchor-block handle (HAB). A return value of 0, however, indicates an error while initializing PrintManager.

Ensure that you save the HAB value that the SPRINIT (Initialize PrintManager) verb returns. You will need to specify this value on other PrintManager verbs. You cannot issue two successive calls to SPRINIT (Initialize PrintManager). You must terminate PrintManager with the SPRTERM (Terminate PrintManager) verb before reinitializing PrintManager.

SPRTERM (Terminate PrintManager)

MVS	VM	OS/400	OS/2
X	X	X	

Function

Terminate PrintManager.

Syntax

```
SPRTERM (hab, success)
```

Figure E-2. Format of the SPRTERM (Terminate PrintManager) Verb

Parameters

hab (HAB)
(Input). The anchor-block handle returned from a successful call to SPRINIT (Initialize PrintManager).

Usage

Use the SPRTERM (Terminate PrintManager) verb to terminate PrintManager and to release all associated resources. For the *hab* parameter, ensure that you enter the anchor-block handle returned on the SPRINIT (Initialize PrintManager) verb when you initialized PrintManager. You must terminate a PrintManager session before you can reinitialize PrintManager.

For the *hab* parameter, ensure that you enter the anchor-block handle returned by the SPRINIT (Initialize PrintManager) verb. Otherwise, in the 370 environment PrintManager will abend with an abend code of X'0245' and in OS/400 a value of 0 will be returned.

Return Values

***success* (BOOL)**
A nonzero value means PrintManager was terminated successfully, whereas a value of 0 means an error occurred.

System-Specific Information for MVS

This section describes the system-specific information needed to implement PrintManager in MVS. PrintManager works only with JES2, R3.1.1 and follow-on releases.

SPRSDOC (Start Document) You cannot specify a "document name" for a print job; therefore, the *DocName* parameter is ignored.

SPREDOC (End Document) The value that is returned upon successful completion is an arbitrary positive number.

If the DATATYPE is MODCAP, AFPDS, or AFPDSLINE, and CC is not set or is set to NO, CC is automatically set to MACHINE.

Interpreting Error Codes in MVS

For error codes with additional message information, the **ERRMSG** structure returns the following:

```
MSG_ID <rtc> <rsc> SYS_SERVICE
```

where:

MSG_ID	Is the message ID for the error
<rtc>	Is the return code from a system service
<rsc>	Is the reason code from a system service
SYS_SERVICE	Is the name of the system service where the error occurred.

For more information on MVS error codes, refer to *MVS/ESA System Programming Library: Application Development Guide.*

System-Specific Information for VM

This section describes the system-specific information needed to implement the PrintManager Interface in VM.

SPRADDF (Add File) Print file IDs passed to the PrintManager Interface are case sensitive.

If you do not specify a file mode, PrintManager searches for the file only on your A disk. If you want to do a global search, specify a file mode of "*".

SPRSDOC (Start Document) System and inline resources are case sensitive.

If you do not specify a file mode, PrintManager searches for the file only on your A disk. If you want to do a global search, specify a file mode of "*".

If the DATATYPE is MODCAP AFPDS, or AFPDSLINE, and CC is not set or is set to NO, CC is automatically set to YES.

Interpreting Error Codes in VM

For some error codes with additional message information, **ERRMSG** returns information in the following format:

```
MSG_ID  cccxxxnnnE  SYS_COMMAND
```

where:

MSG_ID	Is the type of system error that occurred. This will be in the *identifier* field of **ERRMSG** and is defined by PrintManager.
cccxxxnnnE	Is the error message identifier from the system command that failed. This will be in the *name* field of the **ERRMSG** structure.
SYS_COMMAND	Is the system command that failed. This value might be a CP command or a Diagnose instruction. This will be in the *value* field of the **ERRMSG** structure.

The PMERR_PRM_OPEN_ERROR can occur when the SPROPEN (Open) verb or the SPRSDOC (Start Document) verb is issued. This list describes the additional message information you can receive from calls to the error verbs, as follows:

NO_DEV00E	No virtual device 00E is defined on your VM system.
GEN_VADDR_ERROR	The system is unable to generate a new virtual printer address. All possible virtual addresses from X'100' to X'5FF' might be in use.
STORE_00E_ERROR	An error occurred storing your current device 00E.
DEF_VADDR_ERROR	An error occurred defining a new virtual device 00E for this print session.
MULTIPLE_SESSION_ERROR	An error occurred because VM allows only one print session to be open at a time.
SYSTEM_ERROR	An error occurred writing an XAB to the virtual printer or an error occurred issuing the CP SPOOL command.
XAB_SIZE_ERROR	The XAB being built from specified options is too large. Specifying a large number of resources could cause this error, as well as specifying print-option values that are not supported by the system. Maximum size is 32,767 bytes.

For more information on VM error codes, refer to *VM/SP Library Guide, Glossary, and Master Index* and *VM/XA Library Guide, Glossary, and Master Index*

VM Single-Print-Session Support

In VM, you can have only one print session open at a time. Each print session has a new virtual printer defined at address 00E. All of the print jobs within a print session use the same virtual printer device 00E defined for the print session. Virtual printer device 00E has various job-option values associated with it. Therefore, the first print job inherits print job options from virtual printer device 00E. Subsequent print jobs inherit options from previous print jobs in the session. These job-option values can be overridden using the SPRSOPT (Set Option) verb. The print-option values inherited from virtual printer device 00E are as follows:

- CLASS
- COPY

- HOLD/NOHOLD
- FORM
- JOBOWNER (CP SPOOL FOR)
- (CP SPOOL TO)
- DIST
- FLASH
- FLASHCNT
- DEST (PRTNAME in PrintManager)
- CHARS
- MODIFY
- MODIFYTRC
- FCB

Note: You must define a virtual printer device at address 00E before invoking PrintManager, or you will receive an error code from SPROPEN (Open). If you are using a 3800-1 or 3800-3 printer, the following additional attributes are inherited:

- SIZE
- Number of WCGMs
- CFS/BTS
- DATCK/NODATCK

The following additional attributes are inherited with VM/XA:

- EOF/NOEOF
- KEEP/NOKEEP
- MSG/NOMSG
- NONAME/NAME fn ft

PrintManager also inherits TAG information associated with device 00E. If device 00E has print data associated with it, device 00E will be closed first, causing a spool file to be created from the print data. Previously created print data cannot be concatenated with newly created PrintManager print data.

System-Specific Information for OS/400

This section describes the system-specific design needed to implement PrintManager in OS/400.

SPREDOC (End Document) The value that is returned upon successful completion is an arbitrary positive number.

When using PrintManager in an application, verify the user's access to files and resources before calling SPRGERI (Get Error Information) for specific error information.

All object names and filenames are translated to uppercase.

All options that are supported by PrintManager should be specified through PrintManager interfaces. Using overrides can cause unpredictable results.

Interpreting Error Codes in OS/400

For some error codes with additional message information, **ERRMSG** returns information in the following format:

```
MSG_ID  cccaaaa  system text
```

Where:

MSG_ID	Is the type of system error that occurred. This will be in the *identifier* field of **ERRMSG** and is defined by PrintManager.
cccaaaa	Is the error message identifier from the system command that failed. This will be in the *name* field of the **ERRMSG** structure.
system text	Is the error message text associated with the system error identifier.

Printing MO:DCA-P Documents in VM and MVS

The PrintManager Interface invokes two data transforms to convert data streams in the MVS and VM environments. One transform converts a MO:DCA-P data stream of PT1 text and DR2 graphics to an AFPDS data stream with PT1 text and IM1 image. This transform is invoked when the following print options have these values:

Option	Value
DATATYPE	MODCAP
MODCASET	AFPDS
GDDMDEVTOKEN	Any valid GDDM family 4 device type. Refer to *GDDM Base Programming Reference* for a list of valid GDDM device types.

If you intend to use the MO:DCA-P transform, GDDM must be available when you run your print application. In VM, you must be linked to the disk where GDDM is installed. In MVS, you must have GDDM in your program search path.

The second transform uses a MO:DCA-P data stream and places each structured field on record boundaries with X'5A' in the first byte of each record. This transform is invoked when the following print options have these values:

Option	Value
DATATYPE	MODCAP
MODCASET	AFPDSE1

Note: MO:DCA-P records are restricted to a maximum length of 32767 bytes.

Appendix F. Verb Error Codes

This appendix uses the following format to provide information about the message identifiers for the PrintManager Interface verbs:

- Error Codes, which consist of the decimal error value, the hexadecimal error value (in parentheses), and the natural language description of the error code.
- Severity level of the message. For more information on severity levels, see "Using PrintManager Error Verbs" on page 3-6.
- Explanation.
- List of PrintManager Interface verbs that may have issued the error
- Response.
- Error code constants for C and COBOL language. In C Language, these constants will be included in your program if you define INCL_ERRORS and include the ekipmgr.h file as described in "Header Files" on page B-1. In COBOL, these constants will be included in your program if you include the copy file (EKICONC) as described in "Copy Files" on page C-1.
- Additional messages issued (if any). Additional message information includes:
  - Message ID, which consists of the hexadecimal error value and corresponding constant.
  - *Value* string containing additional error information.

    For more information on the parameters that contain these strings, either see Chapter 4, PrintManager Interface Verb Reference for descriptions of the verbs issuing the error codes or see "PrintManager Interface Verb Data Types" on page 4-2 for descriptions of the data structures that contain these parameters.

Note: These error codes have a consistent definition in all supported PrintManager environments, but their occurrence and the additional messages returned may be unique for different environments.

For more information on using the PrintManager Interface verbs for problem diagnosis, see "Handling Problems" on page 3-6.

PrintManager Interface Verb Error Codes

The following errors might be returned when you call the PrintManager Interface verbs.

0000 (X'0000')

No error

Severity: Error

Explanation: No error occurred.

PrintManager Interface Verbs: All.

Response: No response is necessary.

Error Code Constants:

C PMERR_OK
COBOL PMGR-PRM-OK

8452 (X'2104')

SPRSDOC Not Issued

Severity: Error

Explanation: The verb that caused this error cannot be issued if a print job has not been started.

PrintManager Interface Verbs: SPRADDF, SPRADOC, SPREDOC, SPRWRIT

Response: SPRSDOC must be issued before the verb that caused this error. For further information on operating states, see "PrintManager Interface Operating States" on page 3-5.

Error Code Constants:

C PMERR_PRM_STARTDOC_NOT_ISSUED
COBOL PMGR-PRM-STARTDOC-NOT-ISSUED

8459 (X'210B')

SPREDOC Not Issued

Severity: Error

Explanation: The verb that caused this error cannot be issued while a print job is being created.

PrintManager Interface Verbs: SPRCLOS, SPRSOPT, SPRSDOC

Response: SPREDOC must be issued before the verb that caused this error. For further information on operating states, see "PrintManager Interface Operating States" on page 3-5.

Error Code Constants:

C PMERR_PRM_ENDDOC_NOT_ISSUED
COBOL PMGR-PRM-ENDDOC-NOT-ISSUED

16389 (X'4005')

Invalid print session handle

Severity: Error

Explanation: The *hprm* parameter does not identify a valid print session.

PrintManager Interface Verbs: SPRADDF, SPRABRT, SPRADOC, SPRCLOS, SPREDOC, SPRLOPT, SPRSOPT, SPRSDOC, SPRQOPT, SPRWRIT

Response: Ensure that:

- You use SPROPEN to open a print session and return a session identifier.
- You use the session identifier on subsequent PrintManager Interface verb calls that require this identifier.

Error Code Constants:

C PMERR_PRM_INV_HPRM
COBOL PMGR-PRM-INV-HPRM

16391 (X'4007')

Insufficient Storage

Severity: Error

Explanation: Processor storage was not sufficient.

PrintManager Interface Verbs: SPRABRT, SPRADOC, SPRADDF, SPREDOC, SPRLOPT, SPROPEN, SPRQOPT, SPRSOPT, SPRSDOC, SPRWRIT

Response: Either allocate more processor storage or reduce the storage required by the PrintManager Interface application.

Error Code Constants:

C PMERR_PRM_NO_MEMORY
COBOL PMGR-PRM-NO-MEMORY

16392 (X'4008')

Cannot abort spool file

Severity: Error

Explanation: A system error occurred with the spool subsystem while aborting the print job.

PrintManager Interface Verbs: SPRADDF, SPRADOC, SPRABRT, SPREDOC, SPRSDOC, SPRWRIT

Response: Correct the system error indicated by the additional message information and rerun the application. In VM, ensure that printer device 00E is not detached. In MVS, ensure that the SYSOUT data set is not deallocated. In OS/400, check the job log for additional system error information. For VM, see "System-Specific Information for VM" on page E-4 for more information on the additional message information. For MVS, see "System-Specific Information for MVS" on page E-4 for more information on the additional message information.

Error Code Constants:

C PMERR_PRM_PRINT_ABORT
COBOL PMGR-PRM-PRINT-ABORT

Additional Message Information:

Message ID	Value
19712 (X'4D00') system error	System information and return values
19713 (X'4D13') MVS spool file error	System information and return values

16397 (X'400D')

No data on the spool

Severity: Error

Explanation: There is no data on the spool.

PrintManager Interface Verbs: SPREDOC

Response: Ensure that no errors occurred while writing data to the spool.

Error Code Constants:

C PMERR_PRM_NO_DATA
COBOL PMGR-PRM-NO-DATA

16410 (X'401A')

Invalid Length or Count

Severity: Error

Explanation: The length of data to be written to the spool is too long.

PrintManager Interface Verbs: SPRADDF, SPREDOC, SPRSDOC, SPRWRIT

Response: Ensure that the length of your input data does not exceed the length limitations of your system.

Error Code Constants:

C PMERR_PRM_INV_LENGTH_OR_COUNT
COBOL PMGR-PRM-INV-LENGTH-OR-COUNT

Additional Message Information:

Message ID	Value
19712 (X'4D00') system error	System information and return values

16412 (X'401C')

Cannot Open File

Severity: Error

Explanation: An error occurred opening the file.

PrintManager Interface Verbs: SPRADDF

Response: Ensure that the file exists and that it is accessible. Ensure that the file name is spelled correctly.

Error Code Constants:

C PMERR_PRM_CANNOT_OPEN_FILE
COBOL PMGR-PRM-CANNOT-OPEN-FILE

Additional Message Information:

Message ID	Value
19722 (X'4D0A') input file error	file name

16544 (X'40A0')

Cannot close spool file

Severity: Error

Explanation: A system error occurred closing the spool file.

PrintManager Interface Verbs: SPRABRT, SPRCLOS, SPREDOC, SPROPEN

Response: Correct the system error indicated by the additional message information and rerun the application. In VM, ensure that printer device 00E is not detached. In MVS, ensure that the SYSOUT data set is not deallocated. For VM, see "System-Specific Information for VM" on page E-4 for more information on the additional message information. For MVS, see "System-Specific Information for MVS" on page E-4 for more information on the additional message information. In OS/400, check the job log for additional system error information.

Error Code Constants:

C PMERR_PRM_CLOSE_ERROR

COBOL PMGR-PRM-CLOSE-ERROR

Additional Message Information:

Message ID	Value
19712 (X'4D00') system error	System information and return values
19715 (X'4D03') VM restore device 00E error	System information and return values
19731 (X'4D13') MVS spool close error	System information and return values

16545 (X'40A1')

Open Error

Severity: Error

Explanation: A system error occurred opening the spool file.

PrintManager Interface Verbs: SPRSDOC

Response: Correct the system error indicated by the additional message information and rerun the application. Check the job log for additional system error information.

Error Code Constants:

C PMERR_PRM_OPEN_ERROR
COBOL PMGR-PRM-OPEN-ERROR

Additional Message Information:

Message ID	Value
19724 (X'4D0C') MVS spool open error	System information and return values
19729 (X'4D11') MSV OUTADD macro error	System information and return values
19730 (X'4D12') MVS OUTDEL macro error	System information and return values
19716 (X'4D04') VM XAB size error	System information and return values
19713 (X'4D01') VM no device 00E error	System information and return values
19726 (X'4D0E') VM generate virtual address error	System information and return values
19725 (X'4D0D') VM saving user device 00E error	System information and return values
19714 (X'4D02') VM define virtual address error	System information and return values
19728 (X'4D10') VM multiple session error	System information and return values
19712 (X'4D00') system error	System information and return values

16546 (X'40A2')

Invalid Option Value

Severity: Error

Explanation: The specified print-option value is not supported in a FORMDEF or is not supported by the operating system.

PrintManager Interface Verbs: SPRSDOC

Response: Specify a value that satisfies the validation rules and valid values in the current print descriptor. Ensure that the value is valid for the current operating system environment.

Error Code Constants:

C PMERR_PRM_INV_OPTVALUE
COBOL PMGR-PRM-INV-OPTVALUE

Additional Message Information:

Message ID	Value
19723 (X'4D0B') invalid value	Option name, value
19727 (X'4D0F') invalid FORMDEF option	Option name, value
19717 (X'4D05') invalid print option	Option name, value
19737 (X'4D19') binary value	Option name

16547 (X'40A3')

Read Error

Severity: Error

Explanation: An error occurred reading the file.

PrintManager Interface Verbs: SPRADDF

Response: If no hardware error occurred, re-create the file or ensure that the correct file was specified.

Error Code Constants:

C PMERR_PRM_READ_ERROR
COBOL PMGR-PRM-READ-ERROR

Additional Message Information:

Message ID	Value
19722 (X'4D0A') input file error	file name

16548 (X'40A4')

Invalid Resource

Severity: Error

Explanation: The type of resource requested to be placed inline is invalid.

PrintManager Interface Verbs: SPRSDOC

Response: Ensure that the requested inline resource matches the type of resource specified.

Error Code Constants:

C PMERR_PRM_INV_RESOURCE

COBOL PMGR-PRM-INV-RESOURCE

Additional Message Information:

Message ID	Value
19738 (X'4D1A') invalid resource name	Type, resource name

16549 (X'40A5')

Unsupported system

Severity: Error

Explanation: The system is not supported by PrintManager.

PrintManager Interface Verbs: SPROPEN

Response: Contact your system programmer to ensure that PrintManager is installed on a supported system.

Error Code Constants:

C PMERR_PRM_UNSUPPORTED_SYSTEM
COBOL PMGR-PRM-UNSUPPORTED-SYSTEM

16550 (X'40A6')

Cannot write to spool file

Severity: Error

Explanation: A system error occurred while writing print data to the spool.

PrintManager Interface Verbs: SPRADDF, SPREDOC, SPRSDOC, SPRWRIT

Response: Correct the system error indicated by the additional message information and rerun the application. Ensure that the value of the CC print option is the correct value for the print data. For VM, see "System-Specific Information for VM" on page E-4 for more information on the additional message information. For MVS, see "System-Specific Information for MVS" on page E-4 for more information on the additional message information. In OS/400, check the job log for additional system error information.

Error Code Constants:

C PMERR_PRM_WRITE_ERROR
COBOL PMGR-PRM-WRITE-ERROR

Additional Message Information:

Message ID	Value
19712 (X'4D00') system error	System information and return values

16551 (X'40A7')

Option Not Defined

Severity: Error

Explanation: The print option requested could not be found in the set of current options.

PrintManager Interface Verbs: SPRQOPT

Response: Ensure that the print option is set and that the print-option name is spelled correctly.

Error Code Constants:

C PMERR_PRM_OPT_NOT_DEFINED
COBOL PMGR-PRM-OPT-NOT-DEFINED

16552 (X'40A8')

Option Not Set

Severity: Error

Explanation: Specified print options or values are invalid. If a new print descriptor is specified, this error refers to the print options that were set previously but are not valid when validated against the new print descriptor.

PrintManager Interface Verbs: SPRSOPT

Response: Ensure that the specified print-option value satisfies the rules and valid values defined in the current print descriptor.

Error Code Constants:

C PMERR_PRM_OPT_NOT_SET
COBOL PMGR-PRM-OPT-NOT-SET

Additional Message Information:

Message ID	Value
19721 (X'4D09') print option removed	Option name, value (optional)
19720 (X'4D08') invalid value removed	Option name, value
19719 (X'4D07') invalid print option	Option name (optional), value (optional)
19718 (X'4D06') value syntax error	Option name, value
19737 (X'4D19') binary value	Option name

16553 (X'40A9')

No Defined Options

Severity: Error

Explanation: No print options are defined in the set of current options.

PrintManager Interface Verbs: SPRLOPT

Response: Print options must be defined in the set of current options before you issue the SPRLOPT verb.

Error Code Constants:

C PMERR_PRM_NO_PRINT_OPTS
COBOL PMGR-PRM-NO-PRINT-OPTS

16554 (X'40AA')

Buffer Too Small

Severity: Error

Explanation: The buffer size is not large enough to contain the requested information.

PrintManager Interface Verbs: SPRLOPT, SPRSOPT, SPRQOPT

Response: Specify a sufficient buffer size. For more information, see Chapter 4, "PrintManager Interface Verb Reference" on page 4-1.

Error Code Constants:

C PMERR_PRM_BUFFER_TOO_SMALL

COBOL PMGR-PRM-BUFFER-TOO-SMALL

16555 (X'40AB')

Invalid Buffer

Severity: Error

Explanation: No storage was allocated for the buffer.

PrintManager Interface Verbs: SPRLOPT, SPRSOPT, SPRQOPT

Response: Allocate sufficient buffer storage. For more information, see Chapter 4, "PrintManager Interface Verb Reference" on page 4-1.

Error Code Constants:

C PMERR_PRM_INV_BUFFER
COBOL PMGR-PRM-INV-BUFFER

16556 (X'40AC')

Invalid Option Type

Severity: Error

Explanation: The specified print-option value type is not valid.

PrintManager Interface Verbs: SPRSOPT

Response: Specify a valid print-option value type. For more information, see "Print Option Defaults" on page 5-4.

Error Code Constants:

C PMERR_PRM_INV_TYPE
COBOL PMGR-PRM-INV-TYPE

16557 (X'40AD')

Invalid Level

Severity: Error

Explanation: The specified value for the *Level* parameter is not valid.

PrintManager Interface Verbs: SPRLOPT, SPRSOPT, SPRQOPT

Response: Specify a valid value for the *Level* parameter. For more information, see Chapter 4, "PrintManager Interface Verb Reference" on page 4-1.

Error Code Constants:

C PMERR_PRM_INV_LEVEL
COBOL PMGR-PRM-INV-LEVEL

16558 (X'40AE')

Cannot open resource

Severity: Error

Explanation: A system error occurred opening a resource.

PrintManager Interface Verbs: SPRSDOC

Response: Ensure that the resource is available and that you have authority to access it.

Error Code Constants:

C PMERR_PRM_RESOURCE_OPEN_ERROR

COBOL PMGR-PRM-RESOURCE-OPEN-ERROR

Additional Message Information:

Message ID	Value
19738 (X'4D1A') resource name	resource type, resource file name

16559 (X'40AF')

Cannot read resource

Severity: Error

Explanation: A system error occurred reading a resource.

PrintManager Interface Verbs: SPRSDOC

Response: This is a system I/O error. Retry the verb call.

Error Code Constants:

C PMERR_PRM_RESOURCE_READ_ERROR
COBOL PMGR-PRM-READ-ERROR

Additional Message Information:

Message ID	Value
19738 (X'4D1A') resource name	Resource type (optional), resource file name

16560 (X'40B0')

Transform Error

Severity: Error

Explanation: An error occurred transforming data using one of the MO:DCA-P transforms.

PrintManager Interface Verbs: SPRABRT, SPRADOC, SPRADDF, SPREDOC, SPRSDOC, SPRWRIT

Response: Ensure that the input data is in the correct MO:DCA-P format and that the print options are set correctly for requesting the proper transform. See "Printing MO:DCA-P Documents in VM and MVS" on page E-7 for more information on MO:DCA-P transforms.

Error Code Constants:

C PMERR_PRM_TRANSFORM_ERROR
COBOL PMGR-PRM-TRANSFORM-ERROR

Additional Message Information:

Message ID	Value
19735 (X'4D17') close file error	
19719 (X'4D07') no memory	
19732 (X'4D14') open file error	
19733 (X'4D15') write file error	
19724 (X'4D0C') MVS dataset create error	
19734 (X'4D16') GDDM transform error	Transform error information
19736 (X'4D18') delete file error	
19740 (X'4D1C') record error	
19741 (X'4D1D') incomplete structured field error	

16561 (X'40B1')

Option Value Truncated

Severity: Warning

Explanation: For an option of **RANGE**, the specified value has a decimal precision that does not match the decimal precision of the valid value. Therefore, the value was truncated.

PrintManager Interface Verbs: SPROPEN, SPRSOPT

Response: Either:

- Continue processing
- Change the truncated value to a value with the correct precision
- Modify the print descriptor that is used for validation.

Error Code Constants:

C PMERR_PRM_OPTVALUE_TRUNCATED
COBOL PMGR-PRM-OPTVALUE-TRUNCATED

Additional Message Information:

Message ID	Value
19742 (X'4D1E') truncated value saved	saved truncated value

16562 (X'40B2')

Data type for Resource Incorrect

Severity: Error

Explanation: AFP resources have been requested, but the value of the DATATYPE option is not MODCAP, LINE, AFPDS, or AFPDSLINE.

PrintManager Interface Verbs: SPRSDOC

Response: Either specify the correct data type that allows resources, or do not request any resources.

Error Code Constants:

C PMERR_PRM_DATATYPE_FOR_RESOURCE
COBOL PMGR-PRM-DATATYPE-FOR-RESOURCE

Additional Message Information:

Message ID	Value
19739 (X'4D1B') datatype resource mismatch	DATATYPE, requested datatype

16563 (X'40B3')

Invalid Length Parameter

Severity: Error

Explanation: An invalid input length or area length parameter was received by PrintManager.

PrintManager Interface Verbs: SPRADDF, SPRLOPT, SPROPEN, SPRQOPT, SPRSDOC, SPRSOPT

Response: Ensure that the length parameters you are passing are set to valid positive numbers.

Error Code Constants:

C PMERR_PRM_INV_LENGTH_PARAMETER
COBOL PMGR-PRM-INV-LENGTH-PARAMETER

16896 (X'4200')

Invalid anchor block handle

Severity: Error

Explanation: An invalid anchor-block handle was specified on the *hab* parameter, or access to the associated storage was denied. This error code will occur only in OS/400.

PrintManager Interface Verbs: SPROPEN

Response: Specify a valid anchor-block handle or obtain access to the associated storage. For more information, see Chapter 4, "PrintManager Interface Verb Reference" on page 4-1.

Error Code Constants:

C PRTMGR_INV_HAB
COBOL PMGR-INV-HAB

Table F-1. Cross Reference of Extended Error Message IDs to Constants

Value	C Constant	COBOL Constant
19712 (X'4D00')	SYSTEM_ERROR	PMGR-SYSTEM-ERROR
19713 (X'4D01')	NO_DEV00E	PMGR-NO-DEV00E
19714 (X'4D02')	DEF_VADDR_ERROR	PMGR-DEF-VADDR-ERROR
19715 (X'4D03')	RESTORE_00E_ERROR	PMGR-RESTORE-00E-ERROR
19716 (X'4D04')	XAB_SIZE_ERROR	PMGR-XAB-SIZE-ERROR
19717 (X'4D05')	INVALID_OPTION	PMGR-INVALID-OPTION
19718 (X'4D06')	VALUE_SYNTAX_ERROR	PMGR-VALUE-SYNTAX-ERROR
19719 (X'4D07')	INVALID_PRINT_OPTION	PMGR-INVALID-PRINT-OPTION
19720 (X'4D08')	INVALID_VALUE_REMOVED	PMGR-INVALID-VALUE-REMOVED
19721 (X'4D09')	PRINT_OPTION_REMOVED	PMGR-PRINT-OPTION-REMOVED
19722 (X'4D0A')	INPUT_FILE_ERROR	PMGR-INPUT-FILE-ERROR
19723 (X'4D0B')	INVALID_VALUE	PMGR-INVALID-VALUE
19724 (X'4D0C')	DSALLOC_ERROR	PMGR-DSALLOC-ERROR
19725 (X'4D0D')	STORE_00E_ERROR	PMGR-STORE-00E-ERROR
19726 (X'4D0E')	GEN_VADDR_ERROR	PMGR-GEN-VADDR-ERROR
19727 (X'4D0F')	INVALID_FORMDEF_OPTION	PMGR-INVALID-FORMDEF-OPTION
19728 (X'4D10')	MULTIPLE_SESSION_ERROR	PMGR-MULTIPLE-SESSION-ERROR
19729 (X'4D11')	OUTADD_ERROR	PMGR-OUTADD-ERROR
19730 (X'4D12')	OUTDEL_ERROR	PMGR-OUTDEL-ERROR
19731 (X'4D13')	DSUNALLOC_ERROR	PMGR-DSUNALLOC-ERROR
19732 (X'4D14')	OPEN_FILE_ERROR	PMGR-OPEN-FILE-ERROR
19733 (X'4D15')	WRITE_FILE_ERROR	PMGR-WRITE-FILE-ERROR
19734 (X'4D16')	GDDM_TRANSFORM_ERROR	PMGR-GDDM-TRANSFORM-ERROR
19735 (X'4D17')	CLOSE_FILE_ERROR	PMGR-CLOSE-FILE-ERROR
19736 (X'4D18')	DELETE_FILE_ERROR	PMGR-DELETE-FILE-ERROR
19737 (X'4D19')	BINARY_VALUE	PMGR-BINARY-VALUE
19738 (X'4D1A')	RESOURCE_NAME	PMGR-RESOURCE-NAME
19739 (X'4D1B')	DATATYPE_RESOURCE_MISMATCH	PMGR-DATATYPE-RSOURCE-MISMATCH
19740 (X'4D1C')	RECORD_ERROR	PMGR-RECORD-ERROR
19741 (X'4D1D')	INCOMPLETE_SF	PMGR-INCOMPLETE-SF
19742 (X'4D1E')	TRUNCATED_VALUE_SAVED	PMGR-TRUNCATED- VALUE-SAVED

Error Codes from Calls to PrintManager API Verbs

The following errors might be returned when you specify a print descriptor on the SPROPEN or SPRSOPT verb. Refer to *PrintManager Application Programming Interface Reference* for information on these error codes.

Table F-2. Error Codes from Calls to API Verbs

Value	C Constant	COBOL Constant
16642 (X'4102')	PMERR_PRD_INV_VALID_VALUES	PMGR-PRD-INV-VALID-VALUES
16643 (X'4103')	PMERR_PRD_INV_DEFAULT	PMGR-PRD-INV-DEFAULT
16646 (X'4106')	PMERR_PRD7_OPEN_ERROR	PMGR-PRD-OPEN-ERROR
16647 (X'4107')	PMERR_PRD_READ_ERROR	PMGR-PRD-READ-ERROR
16648 (X'4108')	PMERR_PRD_SEEK_ERROR	PMGR-PRD-SEEK-ERROR
16651 (X'410B')	PMERR_PRD_FORMAT_ERROR	PMGR-PRD-FORMAT-ERROR
16652 (X'410C')	PMERR_PRD_BUILD_ERROR	PMGR-PRD-BUILD-ERROR
16653 (X'410D')	PMERR_PRD_GRPALIAS_UNDEFINED	PMGR-PRD-GRPALIAS-UNDEFINED
16654 (X'410E')	PMERR_PRD_GRPLISTPTR_ERROR	PMGR-PRD-GRPLISTPTR-ERROR
16655 (X'410F')	PMERR_PRD_RULE_MISMATCH	PMGR-PRD-RULE-MISMATCH
16656 (X'4110')	PMERR_PRD_BAD_STRING_MERGE	PMGR-PRD-BAD-STRING-MERGE
16657 (X'4111')	PMERR_PRD_BAD_RANGE_MERGE	PMGR-PRD-BAD-RANGE-MERGE
16658 (X'4112')	PMERR_PRD_BAD_LIST_MERGE	PMGR-PRD-BAD-LIST-MERGE
16661 (X'4115')	PMERR_PRD_INV_PRD_NAME	PMGR-PRD-INV-PRD-NAME
16663 (X'4117')	PMERR_PRD_GRPLIST_EMPTY	PMGR-PRD-GRPLIST-EMPTY
16668 (X'411C')	PMERR_PRD_INV_GROUP	PMGR-PRD-INV-GROUP
16669 (X'411D')	PMERR_PRD_NOT_FOUND	PMERR-PRD-NOT-FOUND
16670 (X'411E')	PMERR_PRD_GRPLIST_NOT_FOUND	PMGR-PRD-GRPLIST-NOT-FOUND
16674 (X'4122')	PMERR_PRD_TYPE_MISMATCH	PMGR-PRD-TYPE-MISMATCH
16681 (X'4129')	PMERR_PRD_DEFAULT_TRUNCATED	PMGR-PRD-DEFAULT-TRUNCATED
16682 (X'412A')	PMERR_PRD_VALID_ VALUE_TRUNCATED	PMGR-PRD-VALID- VALUE-TRUNCATED

Appendix G. IBM SAA PrintManager Operating Environment

PrintManager/400 is available in OS/400 Release 3 and Version 2 Release 1. PrintManager Interface C language bindings are available with Release 3, and PrintManager Interface C, COBOL, and RPG bindings are available with Version 2 Release 1.

Machine Requirements

MVS Machine Requirements

IBM SAA PrintManager is supported on all processors that run under MVS/SP Version 3 Release 1.0/e (MVS/ESA) with JES2.

VM Machine Requirements

IBM SAA PrintManager is supported on all processors that run under:

- Virtual Machine Facility/System Product (VM/SP) Release 5 or higher
- Virtual Machine Facility/System Product High Performance Option (VM/SP HPO) Release 5 or higher
- Virtual Machine/Extended Architecture (VM/XA) Release 2 or higher
- VM/ESA Release 1.

Programming Requirements

General Programming Requirements

- IBM SAA PrintManager requires the IBM C/370 Library Version 1 Release 2 (5688-039).

  In addition, the PRF menu interactive interface requires the following:

  - ISPF/PDF Version 3 Release 2 (5685-054) for MVS
  - ISPF Version 3 Release 2 (5684-043) for VM.

- Customer applications that use the PrintManager Interface and API must be written in the C programming language, and they will require the IBM C/370 Compiler Version 1 Release 2 (5688-040).
- Distributing print jobs between host systems (MVS and VM) requires the following:

  - JES2 Version 3 Release 1 Modification 1 (5685-001).
  - RSCS Version 2 Release 2 (with PTF VM30575), or Version 2 Release 3 (5664-188).

- If Print Services Facility (PSF) is used as the printer driver, one of the following is required:

  - PSF/VM Version 1 Release 3 (5664-198) or higher
  - PSF/MVS Version 1 Release 3 (5665-275) or higher.

- One of the following Graphical Data Display Manager products is required when submitting a print job that contains a MO:DCA-P data stream of PT1 text and DR2 graphics that will be printed on a printer that only supports an AFPDS data stream with PT1 text and IM1 image:
  - GDDM/MVS Version 2 Release 3 (5665-356)
  - GDDM/VM Version 2 Release 2 (5664-200) for VM/SP Release 5
  - GDDM/VMXA Version 2 Release 3 (5684-007) for VM/SP Release 6 or VM/XA.

Note: In the VM/XA, VM/ESA, and MVS/ESA environments, IBM SAA PrintManager takes full advantage of 31-bit addressing, and can be installed above 16Mb. However, in these environments, IBM SAA PrintManager does not support application programs that operate in 24-bit addressing mode.

MVS Programming Requirements

- IBM System Modification Program Extended (SMP/E) Version 1 Release 5 (5668-949) is required to install IBM SAA PrintManager.
- MVS/SP Version 3 Release 1.0/e (MVS/ESA) with JES2 Version 3 Release 1 Modification 1 (5685-001) must be installed as the operating system before you can install IBM SAA PrintManager.
- The PRF component of IBM SAA PrintManager requires Time Sharing Option/Extensions (TSO/E) Version 2 Release 1 (5685-025).

Notes:

1. IBM SAA PrintManager supports TSO/E and MVS batch job users *only* in the MVS environment.
2. IBM SAA PrintManager in an MVS/ESA system only supports PrintManager applications operating in 31 bit addressing mode.

VM Programming Requirements

Before you can install IBM SAA PrintManager, the operating system must be at one of the following levels:

- VM/SP Release 5 (5664-167), with the Macro Compatibility Enhancement PTF (VM34760) installed.
- VM/SP Release 6 (5664-167).
- VM/SP HPO Release 5 (5664-173), with the Macro Compatibility Enhancement PTF (VM34760) installed.
- VM/SP HPO Release 6 (5664-173).
- VM/XA SP Release 2 (5664-308), with the APSS PTF (VM34943) installed.
- VM/XA SP Release 2 Modification 1 (5664-308)
- VM/ESA Release 1 (5684-112).

Notes:

1. The PRF interactive menus are not supported on VM/SP Release 5 or VM/SP HPO Release 5.
2. IBM SAA PrintManager in VM/XA or VM/ESA systems only supports PrintManager applications operating in 31 bit addressing mode.
3. IBM SAA PrintManager does not support the CMS Shared File System (SFS).

Glossary

Source Identifiers

This publication includes terms and definitions from:

- The *American National Dictionary for Information Processing Systems,* copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A)[12] after the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Committee (ISO/IEC JTC1/SC1). Definitions of published segments of the vocabularies are identified by the symbol (I)[12] after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/IEC JTC1/SC1 vocabulary subcommittee are identified by the symbol (T)[12] after the definition, indicating final agreement has not yet been reached among participating members.

References

The following cross-references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

Deprecated term for. This indicates that the term should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

See. This refers the reader to multiple-word terms that have the same last word.

See also. This refers the reader to related terms that have a related, but not synonymous, meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the glossary.

Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.

A

Advanced Function Printing (AFP). A group of IBM licensed programs (or, for OS/400, operating system functions) and printers that uses the all-points-addressable concept to print text and graphics on a page printer.

Advanced Function Printing data stream (AFPDS). A subset of the Mixed Object Content Document Architecture that consists of the structure fields that Print Services Facility accepts.

AFP. Advanced Function Printing

AFPDS. Advanced Function Printing data stream

AFP resources. AFP resources consist of form definitions, page definitions, fonts, overlays (electronic forms), and page segments (graphic images). With PrintManager, resources can either exist in a system library, or be placed inline with a print job as the job is written to the spool.

API. Application Programming Interface

API verbs. A set of programming verbs used to invoke the PrintManager API functions.

Application Programming Interface. A PrintManager component that can be used to create common print descriptors for an entire organization.

B

burst. To separate continuous-forms paper into separate sheets.

C

Common Programming Interface (CPI). The SAA Common Programming Interface

D

dynamic print management. Using PrintManager to make changes to a print operation without interrupting system functions.

E

electronic overlay. An AFP resource that is a collection of predefined data such as lines, shading, text, boxes, or logos, that can be merged with variable data on a page while printing.

F

font. *1.* A family or assortment of characters of a given size and style; for example, 9 point Bodoni modern.(A)

2. An AFP resource object that defines the fonts for a print job.

form definition. An AFP resource that defines the characteristics of the printed media, for example: overlays to be used, text suppression, position of page data on the form, and number and modifications of a page.

G

GDDM. A group of routines that allows pictures to be defined and displayed procedurally through function routines that correspond to graphic primitives.

I

IBM SAA PrintManager. The PrintManager licensed program for the MVS and VM operating environments.

J

JES. Job Entry Subsystem

Job Entry Subsystem. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system.

M

Mixed Object Document Content Architecture for Presentation. The presentation subset of an architected, device independent data stream for interchanging documents.

MO:DCA-P. See Mixed Object Document Content Architecture for Presentation Interchange Set

O

Operating System/2. An IBM licensed program that can be used as the operating system for the PS/2 processor series.

Operating System/400. An IBM licensed program that can be used as the operating system for the AS/400 processor series.

OS/2. Operating System/2

OS/400. Operating System/400

overlay. See electronic overlay

P

page definition. An AFP resource that contains the formatting controls for line data. It can include controls for number of lines per logical page, font selection, print direction, and mapping individual fields to positions on the logical page.

page segment. An AFP resource that can contain text and images and can be positioned on any addressable point on a page or an electronic overlay.

PRF. See Print Request Facility

print descriptor. A PrintManager object that is created and maintained with the API. Print descriptors for printing describe where a print job will be printed, how it will be processed, and how the output will appear. These print descriptors contain capabilities and defaults of options used for printing. Multiple print descriptors can exist for different devices and types of print jobs, which allows print jobs to be tailored for different types of applications and routed to the correct destination.

PrintManager Interface. An element of the IBM SAA common programming interface that allows you to write portable applications for sending print files to a system spool for printing. PrintManager implements the PrintManager Interface via a set of functions, known as verbs, and the PrintManager print options.

PrintManager. The collective name for a set of IBM programs or operating system functions that provide cross system print management for an entire organization.

PrdT. See Prd Tool

Prd Tool. A set of tags and commands used to create and maintain print descriptors.

Print Request Facility (PRF). A PrintManager component that provides the casual user with a consistent way to submit print jobs.

Print Services Facility. The collective name for a group of IBM licensed programs (or operating system functions, for OS/400) that produce printer commands for page printers.

PSF. Print Services Facility

R

Remote Spooling Communication Subsystem. The licensed program that transfers spool files, commands, and messages between VM users, remote stations, and remote and local batch systems through HASP-compatible telecommunication facilities.

RSCS. Remote Spooling Communication Subsystem

S

SAA. Systems Application Architecture

Set of Current Options. The set of current options contains the print options associated with the current print job, as well as a combination of the default values inherited from the print descriptor and the print-option values overridden by calls SPRSOPT (Set Option).

Systems Application Architecture. A set of software interfaces, conventions, and protocols that provide a framework for designing and developing applications with cross-system consistency.

Index

Q

R

S

T

V

Reader's Comments

Systems Application Architecture
Common Programming Interface
PrintManager Reference

Publication No. S544-3698-00

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

	Yes	No
• Does the publication meet your needs?	☐	☐
• Did you find the information:		
Accurate?	☐	☐
Easy to read and understand?	☐	☐
Easy to retrieve?	☐	☐
Organized for convenient use?	☐	☐
Legible?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐
• Do you use this publication:		
As an introduction to the subject?	☐	☐
As a reference manual?	☐	☐
As an instructor in class?	☐	☐
As a student in class?	☐	☐
• What is your occupation?		

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:

Name

Address

Company or Organization

Phone No.

Reader's Comments
S544-3698-00

IBM®

Cu
Al

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, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191

Fold and Tape | **Please do not staple** | Fold and Tape

Cut
Alo

Reader's Comments

Systems Application Architecture
Common Programming Interface
PrintManager Reference

Publication No. S544-3698-00

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

	Yes	No
• Does the publication meet your needs?	☐	☐
• Did you find the information:		
Accurate?	☐	☐
Easy to read and understand?	☐	☐
Easy to retrieve?	☐	☐
Organized for convenient use?	☐	☐
Legible?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐
• Do you use this publication:		
As an introduction to the subject?	☐	☐
As a reference manual?	☐	☐
As an instructor in class?	☐	☐
As a student in class?	☐	☐
• What is your occupation?		

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:

Name

Address

Company or Organization

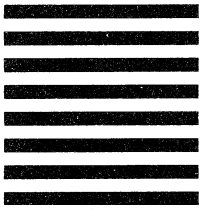
Phone No.

Reader's Comments
S544-3698-00

Cut
Alo

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, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191

Fold and Tape | **Please do not staple** | Fold and Tape

Cut
Alo

File Number: S370-40

Printed in Denmark by FairPrint

Systems Application Architecture Library

GC26-4341	Overview
GC26-4675	Common Programming Interface: Summary
GC31-6810	Common Communications Support: Summary
SC26-4582	Common User Access: Advanced Interface Design Guide
SC26-4583	Common User Access: Basic Interface Design Guide
SC26-4362	Writing Applications: A Design Guide
GC26-4531	AD/Cycle Concepts
GC23-0576	An Introduction to SystemView

Common Programming Interface:

SC26-4355	Application Generator Reference
SC09-1308	C Reference – Level 2
SC26-4354	COBOL Reference
SC26-4399	Communications Reference
SC26-4348	Database Reference
SC26-4798	Database Level 2 Reference
SC26-4356	Dialog Reference
SC26-4357	FORTRAN Reference
SC26-4381	PL/I Reference
SC26-4359	Presentation Reference
S544-3698	PrintManager Reference
SC26-4358	Procedures Language Reference
SC24-5549	Procedures Language Level 2 Reference
SC26-4349	Query Reference
SC26-4684	Repository Reference
SC31-6821	Resource Recovery Reference
SC09-1286	RPG Reference

S544-3698-00